

Copyright
by
Han Hee Song
2011

The Dissertation Committee for Han Hee Song
certifies that this is the approved version of the following dissertation:

Large-Scale Network Analytics

Committee:

Yin Zhang, Supervisor

James Browne

Inderjit Dhillon

Lili Qiu

Jia Wang

Large-Scale Network Analytics

by

Han Hee Song, B.S.; M.A.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2011

To my family.

Acknowledgments

First and foremost, I would like to record my gratitude to my advisors, Prof. Yin Zhang and Prof. Lili Qiu, who guided my masters and doctoral studies to a successful completion. Yin provided me unflinching support and encouragement throughout my study. His remarkable insights had greatly inspired me and enriched my growth as a student and a researcher I wanted to be. Lili has been of great help not only for wireless network researches but also for all of my publications we coauthored. I hope to continue our collaboration in the future.

I gratefully acknowledge Prof. Yan Chen (Northwestern University) for introducing me to this fascinating field of network research and guiding through my first research work.

I also give my deep appreciations to Prof. Inderjit Dhillon, Prof. James Browne, and Dr. Jia Wang (AT&T Labs - Research) for supervising me in collaboration works as well as serving on my dissertation committee. Dr. Dhillon has led me into the extraordinary world of data mining and taught me how to use the mining tools. Dr. Browne has patiently went through my dissertation proposal and thesis, and provided me a lot of insightful ideas shaping up this thesis. And Dr. Wang has showed me the excitement of mixing theory and practice by inviting me to an year-long internship at AT&T Labs.

It is also a pleasure to pay tribute to my other mentors at AT&T labs, Dr. Zihui Ge, Dr. Jennifer Yates, Dr. Andrea Basso, and Dr. Min Zhang.

I am benefited by my collaborators and friends who patiently entertained my research questions: Dr. Tae Won Cho, Dr. Berkant Savas, Dr. Upendra Shevade, Dr. Ajay Mahimkar, Dr. Zhengdong Lu, Vacha Dave, Dr. Mikyung Han, Dr. Eric Rozner, Dr. Tianji Li, Apurv Bhalta, Wei Dong, Sang Min Lee, You Suk Seung, Gene Moo Lee, Ed Wong, Prince Mahajan, David Rager, and all LASReans. Many thanks to their constructive comments on my works as well as encouragements in hard times.

None of this could have been achieved without the dedication of my family. My father, Prof. Bang-ho Song, is the person who put the foundation of my learning character in my DNA. My mother, Prof. Sook-ja Lee, is the one who has shown me the joy of reading and positive thinking. I would like to recognize my uncle, Prof. Dong-ho Song, for initially getting me interested in the amazing field of computer science. I also wish to thank Young-do and Min-chul for being supportive and caring brothers. And I give my special thanks to my dearest wife Myung Kyung for her inherent dedication, love, and persistent confidence in me. Naturally, my deep appreciation goes to my parents-in-law, Seung-bum Chung and Ok-hee Seo, for their support in me forming a family with Myung Kyung.

Last, but certainly not least, I would like to thank the newest member of my family, princess Natalie, for being healthy and happy, and implicitly encouraging me to push hard for completion of my study.

Large-Scale Network Analytics

Han Hee Song, Ph.D.

The University of Texas at Austin, 2011

Supervisor: Yin Zhang

Scalable and accurate analysis of networks is essential to a wide variety of existing and emerging network systems. Specifically, network measurement and analysis helps to understand networks, improve existing services, and enable new data-mining applications. To support various services and applications in large-scale networks, network analytics must address the following challenges: (i) how to conduct scalable analysis in networks with a large number of nodes and links, (ii) how to flexibly accommodate various objectives from different administrative tasks, (iii) and how to cope with the dynamic changes in the networks.

This dissertation presents novel path analysis schemes that effectively address the above challenges in analyzing pair-wise relationships among networked entities. In doing so, we make the following three major contributions to large-scale IP networks, social networks, and application service networks.

For IP networks, we propose an accurate and flexible framework for path property monitoring. Analyzing the performance side of paths between

pairs of nodes, our framework incorporates approaches that perform exact reconstruction of path properties as well as approximate reconstruction. Our framework is highly scalable to design measurement experiments that span thousands of routers and end hosts. It is also flexible to accommodate a variety of design requirements.

For social networks, we present scalable and accurate graph embedding schemes. Aimed at analyzing the pair-wise relationships of social network users, we present three dimensionality reduction schemes leveraging matrix factorization, count-min sketch, and graph clustering paired with spectral graph embedding. As concrete applications showing the practical value of our schemes, we apply them to the important social analysis tasks of proximity estimation, missing link inference, and link prediction. The results clearly demonstrate the accuracy, scalability, and flexibility of our schemes for analyzing social networks with millions of nodes and tens of millions of links.

For application service networks, we provide a proactive service quality assessment scheme. Analyzing the relationship between the satisfaction level of subscribers of an IPTV service and network performance indicators, our proposed scheme proactively (*i.e.*, detect issues before IPTV subscribers complain) assesses user-perceived service quality using performance metrics collected from the network. From our evaluation using network data collected from a commercial IPTV service provider, we show that our scheme is able to predict 60% of the service problems that are complained by customers with only 0.1% of false positives.

Table of Contents

Acknowledgments	v
Abstract	vii
List of Tables	xiii
List of Figures	xiv
Chapter 1. Introduction	1
1.1 Challenges	3
1.2 Research	5
1.2.1 Accurate and Flexible Inference of Network Properties .	6
1.2.2 Scalable and Accurate Social Network Analysis	7
1.2.3 Proactive Service Quality Assessment	9
1.3 Outline	10
Chapter 2. Accurate and Flexible Inference of Network Properties	11
2.1 Problem Formulation	16
2.2 Exact Reconstruction of Network Path Properties	19
2.2.1 Algebraic Model	19
2.2.2 Basic Static Algorithms	22
2.2.2.1 Measurement Paths Selection	22
2.2.2.2 Path Loss Rate Calculations	23
2.2.3 Scalability Analysis	25
2.3 Approximate Reconstruction of Network Path Properties . . .	29
2.3.1 Design of Experiments	29
2.3.1.1 Bayesian Experimental Design	30
2.3.1.2 Flexibility	38

2.3.1.3	Non-Bayesian Designs	40
2.3.2	Inference Algorithms	43
2.3.2.1	L_2 Norm Minimization	45
2.3.2.2	L_1 Norm Minimization	45
2.3.2.3	Maximum Entropy Estimation	47
2.3.3	Toolkit Development	47
2.3.4	Evaluation	49
2.3.4.1	Accuracy Metric	49
2.3.4.2	Dataset Description	50
2.3.4.3	Experimental Parameters	53
2.3.4.4	Basic Framework Evaluation Results	54
2.3.4.5	Flexibility of Measurement Design	62
2.3.4.6	Augmented Design	63
2.3.5	Effects of Prior Information	68
2.4	Summary of Network Property Inference	69
Chapter 3.	Scalable and Accurate Social Network Analysis	70
3.1	Problem Formulation	74
3.1.1	Proximity measures	74
3.2	Scalable Proximity Inversion	77
3.2.1	Preparation	77
3.2.2	Proximity Sketch	80
3.2.3	Proximity Embedding	82
3.2.4	Evaluation	84
3.2.4.1	Dataset Description	84
3.2.4.2	Accuracy Evaluation of Proximity Embedding	86
3.2.4.3	Accuracy Evaluation of Proximity Sketch	90
3.2.4.4	Scalability	91
3.3	Clustered Spectral Graph Embedding	95
3.3.1	Proposed Methodology	95
3.3.1.1	Basic algorithm	98
3.3.1.2	Advantages	100

3.3.1.3	Extensions	105
3.3.2	Applications	108
3.3.2.1	Proximity estimation	108
3.3.2.2	Missing link inference	109
3.3.2.3	Link prediction	110
3.3.2.4	Supervised learning	113
3.3.3	Evaluation	115
3.3.3.1	Dataset description	116
3.3.3.2	Scalability	118
3.3.3.3	Proximity measure estimation	123
3.3.3.4	Missing link inference	126
3.3.3.5	Link prediction evaluation	127
3.4	Summary of Social Network Analysis	131
Chapter 4.	Proactive Service Quality Assessment	132
4.1	Background	137
4.1.1	IPTV Service Architecture	137
4.1.2	Data Sets	138
4.2	Q-score Design	142
4.2.1	Overview	142
4.2.2	Spatio-Temporal Feature Extraction	144
4.2.2.1	Transformation of Measurement Readings	144
4.2.2.2	Construction of Measurement Matrix	147
4.2.2.3	Multi-scale Temporal Level Aggregation	148
4.2.2.4	Multi-scale Spatial Level Aggregation	149
4.2.3	Feedback Aggregation	150
4.2.4	Regression	151
4.2.5	Compute Q-score in Runtime	153
4.3	Evaluation	154
4.3.1	Evaluation Methodology	154
4.3.2	Results	155
4.3.2.1	Accuracy Analysis	155

4.3.2.2	Multi-scale Temporal Aggregation	157
4.3.2.3	Multi-scale Spatial Aggregation	158
4.3.2.4	Feedback Aggregation	160
4.3.2.5	Sensitivity to Training Duration	161
4.4	Application	162
4.4.1	Identification of Significant KPIs	163
4.4.2	Predicting Bad Quality of Experience	167
4.4.3	Dimensioning Customer Care Workforce	169
4.5	Summary of Proactive Service Quality Assessment	172
Chapter 5.	Conclusion	174
5.1	Contributions	174
5.1.1	Accurate and Flexible Inference of Network Properties .	174
5.1.2	Scalable and Accurate Social Network Analysis	175
5.1.3	Proactive Service Quality Assessment	177
5.2	Future Works	177
Bibliography		180
Vita		189

List of Tables

2.1	Table of notations	16
2.2	Additional table of notations	20
2.3	Inference algorithms	44
2.4	Summary of PlanetLab traces used for evaluation.	50
2.5	Summary of Rocketfuel topologies used for evaluation.	51
2.6	Summary of BRITE topologies used for evaluation.	53
3.1	Dataset summary	85
3.2	Comparing proximity embedding and proximity sketch in estimating large Katz values.	88
3.3	Comparing proximity embedding and proximity sketch in estimating large RPR values.	89
3.4	Computation time of proximity embedding, proximity sketch, common neighbor and shortest path distance.	91
3.5	Description of the proximity measures.	109
3.6	Summary of the online social network datasets.	116
3.7	Clustering results with Graculus. c is the number of clusters and μ is the average cluster size.	118
3.8	Comparison of preparation times ($c = 20$ and $r = r_i = 100$). All timings are in minutes.	119
3.9	Comparison of query times ($r = r_i = 100$, 0.6 million samples).	119
3.10	Computational time and memory usage for the CSGE. Network size m and number of links are approximate.	122
4.1	Summary of Data sets	139
4.2	Accuracy analysis results of Q-score	157
4.3	Training and testing durations	162
4.4	Significant KPIs for large δ (15-24 hrs)	165
4.5	Significant KPIs for small δ (3-9 hrs)	165
4.6	Significant KPIs for multi-scale temporal aggregation (0-24 hrs)	166

List of Figures

2.1	Matrix size representations.	20
2.2	Sample overlay network.	21
2.3	Path (row) selection algorithm	23
2.4	Regression of k in various functions of n under different router-level topologies.	24
2.5	Sequential path selection for Bayesian designs.	35
2.6	SVD based path selection algorithm	42
2.7	QR based path selection algorithm	43
2.8	Our toolkit architecture.	48
2.9	Comparison of inference algorithms for delay and loss estimation using A-optimal design.	55
2.10	Comparison of experimental design schemes for estimating network-wide delay using MinL2 inference algorithm.	56
2.11	Comparison of experimental design schemes for estimating network-wide delay using MinL1_nonNeg inference algorithm.	56
2.12	Comparison of experimental design schemes for per-path delay inference using MinL1_nonNeg inference algorithm.	57
2.13	Comparison of experimental design schemes for per-path loss inference using MinL1_nonNeg inference algorithm (simulation uses Gilbert model).	57
2.14	Comparison of experimental designs for estimating network-wide delay.	59
2.15	False positive and false negative rates of per-path loss inference using MinL1_nonNeg inference algorithm in PlanetLab-Loss.	60
2.16	Comparison of experimental design schemes for per-path delay inference using MinL1_nonNeg inference algorithm on large BRITE topologies.	61
2.17	Use differentiated design based on A-optimal design to provide a higher resolution in estimating a subset of preferred paths in PlanetLab-RTT.	63
2.18	Comparison of augmented design schemes in PlanetLab-RTT.	64

2.19	Comparison of different design modes in handling multi-user scenarios using PlanetLab-RTT, where all modes use the A -optimal design.	66
2.20	Comparison of different design schemes using their best modes on PlanetLab-RTT.	67
2.21	Effects of the enhanced prior on PlanetLab-RTT.	68
3.1	Proximity sketch	79
3.2	Proximity embedding	83
3.3	Fraction of total variance captured by the best rank- k approximation to inter-landmark proximity matrices $\text{Katz}[L, L]$ and $\text{RPR}[L, L]$	93
3.4	Relative errors for top 1%, 5%, and 10% largest values (Katz measure, $\beta = 0.05$, 1600 landmarks, and 60 dimensions). . . .	94
3.5	Sparsity pattern of an adjacency matrix with $c = 10$ clusters. 80% edges are within diagonal blocks.	99
3.6	Illustration of the regular SGE $A \approx UAU^T$ and CSGE $A \approx VSV^T$ in (Eq. 3.19) with $c = 3$ clusters.	100
3.7	Shortest path distance of positive sets \mathcal{P}_1	117
3.8	Comparison of eigen decomposition times.	121
3.9	Accuracy of proximity estimation for LiveJournal dataset. . . .	126
3.10	Missing link inference accuracy for different proximity measures. .	127
3.11	Link prediction accuracy of different measures.	129
3.12	Link prediction accuracy with 2-hop scenario.	130
4.1	IPTV service architecture	138
4.2	Overview of Q-score design	143
4.3	Comparison of various δ s on features (performance indicators) .	158
4.4	Comparison of multi-scale temporal aggregations on features (performance indicators)	159
4.5	Comparison of various spatial aggregation levels on features (performance indicators)	160
4.6	Comparison of various time bin size γ on user feedback	160
4.7	Comparison of accuracy with various training durations	161
4.8	Growth pattern of Q-score	168
4.9	Comparison of accuracy of Q-scores with different skip intervals	169

4.10	Percentage of users with trouble tickets over different spatial locations (COs) and times.	171
4.11	Trend of customer complaints and Q-score per CO.	172

Chapter 1

Introduction

The explosive growth of today's networks has created exciting research opportunities in analytics. The analysis of the rich information embodied in networks leads to a plethora of potential applications for network security (*e.g.*, fighting spam[37], defending against Sybil attacks[106, 100]), systems research (*e.g.*, overlay networks[4, 83], content delivery systems[102, 1]), information technology (*e.g.*, improving Internet search[67] and content recommendation[9]), business (*e.g.*, fraud detection[22], viral marketing[45]), etc.

Despite the significant benefits obtainable from networks, understanding today's networks is not a trivial task. One reason is the increasing volume and diversity of networks. Today's ISPs and enterprise networks can have millions of overlay paths; commercial service networks and online social networks can serve millions of user nodes. At the same time, a wide variety of networks have been introduced on IP networks — from the traditional World Wide Web (WWW) to the newly emerging Online Social Networks (OSNs) and content streaming networks (Internet Protocol TV, Voice over IP, and teleconference networks). With such increasing volume and diversity, it is

difficult to understand even the very basic structure and performance of networks. The other reason that analyzing networks is difficult is the increasing demand for performance-aware applications. Users of networked games and Rich Internet Applications (RIAs) on mobile phones are sensitive to Quality of Service (QoS) and require for better Quality of Experience (QoE). With a large body of performance sensitive applications demanding better performance and higher reliability, it is difficult to monitor and improve service quality.

Given such highly sophisticated network systems and diverse user requirements, network analytics can provide several benefits: First, they can help us to understand networks. Second, they can be used to improve the quality of network services. And third, they enable new data mining applications.

Understanding networks. Measurement of node relationship leads to an understanding of network structure. Monitoring traffic from each network entity further allows an understanding of dynamically changing link and path performances represented as delay, loss-rate, bandwidth, and etc. These findings can in turn be used for predicting trend changes in the near future.

Improving service quality. Leveraging the measured data as input, we can conduct network management and traffic engineering for emerging services with high requirements for QoS and QoE. Such comprehensive service level monitoring can allow quality sensitive services to support the rapidly growing number of subscribers while maintaining reliable service.

Enabling new data mining applications. By further mining and analyzing

the measured data, we can discover new applications. Estimating the proximity (*e.g.*, closeness or similarity) among users of OSNs, for example, forms the basis for enabling a wide range of applications ranging from systems research (*e.g.*, constructing socially aware overlay networks [79]) to evolutionary study in natural sciences (*e.g.*, modeling complex networks [47, 8, 24, 74]).

1.1 Challenges

While network analytics can provide important benefits, analyzing large-scale networks creates the following significant research challenges:

- **Scalability in network analysis space**

As networks evolve to gather more users and subscribers and support more functionalities, the size of networks rapidly increases over time. Thus, an analysis must efficiently handle the workloads of these massive and ever-growing networks. In a network with $O(N)$ nodes where N is the number of nodes in the network, the number of paths interconnecting the nodes can be as many as $O(N^2)$. The quadratic growth in the number of network paths with respect to the number of network nodes makes it impractical to measure every path.

Using the brute force way of simultaneously measuring all network paths — from each end node to all other end nodes, in a large networked system — can lead to inaccurate measurement results and pose an enormous load on node and network resources. A naive scheduling solution that schedules a few measurements at a time on each node can reduce the loads on any

given node and network and avoid interference with other processes and network flows. The increased measurement time, however, can result in stale, inaccurate measurement results.

- **Flexibility in accommodating multiple analysis objectives and constraints**

Different applications require different metrics to be analyzed. The design space is often large, involving a number of control variables. In an experiment measuring the path performance of an IP network, control variables involves choosing the sites to launch experiments from, choosing the type of network characteristics to measure, choosing how to randomize, choosing the granularity, frequency, and duration of each experiment, and so forth.

In addition to the various design choices, there exist multiple design objectives. Different applications (*e.g.*, VoIP gateway services versus cable access network providers) often have different notions of performance, and want to analyze different metrics (*e.g.*, delay or loss in the case of IP networks, and bandwidth and jitter in IPTV service networks).

Finally, the design is often subject to all kinds of constraints imposed by the network and operations, such as the accessibility of measurement infrastructure, the availability of network resources, and operational policies and restrictions.

- **Robustness to the dynamic nature of network**

Today's network environments are highly dynamic: path performance varies

over time, links flap, and new nodes are continually added and deleted. In order to keep track of the changing conditions in real-time, network measurement should work in a timely fashion. Specifically, the measurement of performance metrics should offer fast adaptation to topology changes and balanced distribution of loads to measurement hosts.

When processing the measured data, analytics should incorporate efficient computation such as partial, incremental updates to overcome the overhead of computing the entire network snapshot from scratch.

Even if the measurement is performed in the best way possible, due to the massive scale of the measurement space, there is no guarantee that the measured data will be perfect. Large-scale analytics should be designed to yield the maximum accuracy even when the measurement results are noisy and incomplete.

1.2 Research

We develop novel network analytics that effectively address the above challenges under different types of large-scale networks. In doing so, we consider three analyses conducted on IP networks, social networks, and application service networks. In all three large-scale networks, the single issue is the scalability of *path* analysis because the growth of the network path count is quadratic to the growth of the node count. Therefore, in high level, we focus on pairwise relationship among large number of entities.

1.2.1 Accurate and Flexible Inference of Network Properties

For IP networks, we analyze the performance side of paths between pairs of nodes. Here, we design and implement a framework for large-scale path performance measurement and analysis in IP networks. We develop two approaches that aims at (i) exact reconstruction of network path properties and (ii) approximate reconstruction of path properties.

In *exact reconstruction of network path properties*, we devise a network monitoring system that only requires a small subset of paths in order to rebuild the *exact* properties of all the paths in the network. Observing that a large number of network paths are linearly dependent on other paths, we select the linearly independent set as a basis set running active probing. For an overlay network of N nodes, the linear algebraic approach we develop bounds the number of measurements to $O(N \log N)$ while the existing *general metric* systems [5] require $O(N^2)$ measurements.

Once we set up an upper-bound of the monitoring cost this way, in *approximate reconstruction of network path properties*, we further deal with the problem of scalability and flexibility. Here, we present a Bayesian experimental design based framework, *NetQuest*, that provides far better scalability in continuous monitoring in exchange of a minimal sacrifice of accuracy. The framework not only allows us to conduct experiments that span thousands of routers and end hosts, but it also provides better flexibility to accommodate different design requirements of different users: *differentiated design* for providing higher resolution in measuring certain subnets, *augmented design* for

conducting additional measurements given existing observations, and *joint design* for supporting multiple users interested in different parts of the network. The evaluation on the Internet showed that our measurement framework could estimate path delay within 15% error by monitoring less than 2% of the total paths.

1.2.2 Scalable and Accurate Social Network Analysis

For massive-scale social networks, we analyze pair-wise relationship of users by proposing techniques for the measurement and analysis of social structures that quantifies the proximity (*i.e.*, closeness or similarity) between nodes (*i.e.*, users) in the networks. We consider two scalable and flexible proximity approximation works: (i) scalable proximity inversion and (ii) clustered spectral graph embedding.

As a first step, in *scalable proximity inversion*, we estimate a limited sub-family of node proximity measures in social networks with millions of nodes through two dimensionality reduction techniques, *proximity embedding* and *proximity sketch*. In proximity embedding, we compute true proximity measures among only 0.1% of the millions of nodes and derive the proximity structure of entire networks via matrix factorization. In the other dimensionality approach, proximity sketch, we use count-min sketch to exploit the mice-elephant phenomenon and concisely represent network structure. In a performance evaluation using five popular real online social networks, the techniques accurately approximated proximity measures that have a low-rank

structure, such as Katz measure.

While scalable proximity inversion made significant progress in approximating proximity measures that form low-rank matrices, there was still room for improvement to handle proximity measures without low-rankness. As a follow-up, we propose *clustered spectral graph embedding (CSGE)*, which dramatically improve the accuracy and scalability of proximity estimation over proximity embedding and proximity sketch. By combining the concepts of spectral graph embedding and graph clustering, CSGE embeds the massive original graph into a much smaller reduced graph with relatively high dimensionality. While preserving the essential clustering and spectral information of the original graph, the reduced graph can be flexibly used for a variety of analysis tasks instead of the original.

As concrete applications showing the practical value of CSGE, we apply it to three important social analysis tasks: proximity estimation, missing link inference (*i.e.*, inferring the locations of unobserved links based on observed links), and link prediction (*i.e.*, predicting future friendship relationships). In proximity estimation, CSGE not only provide an order of magnitude faster computation than the state-of-the-art technique, but also support approximations with an order of higher rank with an equal amount of memory footprint. The improved computation and memory complexity led CSGE to approximate a wider range of proximity measures (*e.g.*, approximate rooted PageRank measure with 1% of error). In missing link inference, CSGE result in several-fold reduction of the false positive rate subject to the same false negative rate.

In link prediction, it yield up-to a 20% lower false positive rate. The results clearly demonstrate the effectiveness of the approach and serve as the basis for social network applications and evolution models.

1.2.3 Proactive Service Quality Assessment

For application service networks, we examine the relationship between the satisfaction level of subscribers of the service and network performance indicators. Among many types of application service networks, we especially target IPTV services, and present a service quality assessment framework, *Q-score*, which accurately learns a small set of performance indicators most relevant to user-perceived service quality, and proactively infers service quality in a single score.

We evaluate Q-score using network data collected from a commercial IPTV service provider and show that Q-score is able to predict 60% of service problems that are complained by customers with 0.1% of false positives. Through Q-score, we have (i) gained insight into various types of service problems causing user dissatisfaction including why users tend to react promptly to sound issues while late to video issues; (ii) identified and quantified the opportunity to proactively detect the service quality degradation of individual customers before severe performance impact occurs; and (iii) observed possibility to adaptively allocate customer care workforce to potentially troubling service areas.

1.3 Outline

The remainder of this dissertation is organized as follows. Chapter 2 presents our two network monitoring frameworks for accurate and flexible inference of IP network properties. Chapter 3 describes our two proximity approximation techniques for scalable and accurate social network analysis. Then Chapter 4 discusses our novel method of proactive service quality assessment. Chapter 5 concludes with a summary of our major contributions and avenues for future research.

Chapter 2

Accurate and Flexible Inference of Network Properties

The Internet connects over 200 million hosts and nearly a billion users worldwide. How to better understand and thereby control this enormous, decentralized, and constantly evolving infrastructure is a major challenge faced by today's researchers, engineers, and network operators. As an essential means of achieving better understanding, network measurement has become crucial to a variety of existing and emerging network applications, such as ISP performance management, traffic engineering, content distribution, overlay routing, and peer-to-peer applications. For example, ISPs and enterprise networks put increased focus on network performance, and demand capabilities for detailed performance measurement in networks of hundreds or even thousands of nodes. Performance monitoring also becomes a critical capability that allows overlays and peer-to-peer networks to detect and react to changing network conditions.

While much progress has been made in network measurement, two significant challenges remain. First, large-scale network management applications often require the ability to efficiently monitor the whole network. The

quadratic growth in the number of network paths with respect to the number of network nodes makes it impractical to measure every path. Second, existing techniques are often tailored to specific application needs, and thus lack the flexibility to accommodate applications with different requirements.

To address these challenges, we propose two approaches towards the development of accurate and flexible network measurement: (i) *exact reconstruction of network path properties* and (ii) *approximate reconstruction of network path properties*.

Exact Reconstruction of Network Path Properties.

In *exact reconstruction of network path properties* [16] (Section 2.2), we develop a scalable overlay loss rate monitoring system which provides the best accuracy achievable by selecting a minimal subset of paths to monitor so that the loss rates and latencies of all other paths can be inferred. In our proposed technique, we selectively monitor a *basis set* of k paths. Any end-to-end path can be written as a unique linear combination of paths in the basis set. Consequently, by monitoring loss rates for the paths in the basis set, we infer loss rates for all end-to-end paths. This can also be extended to other additive metrics, such as latency. The end-to-end path loss rates can be computed even when the paths contain *unidentifiable links* for which loss rates cannot be computed.

Given the measurements for paths corresponding to the basis set, this linear algebraic scheme can reconstruct the end-to-end performance on all

paths *exactly*. Moreover, we show that the size of the basis set(k) grows as $O(N \log N)$ through linear regression tests. on various synthetic and real topologies. We also provide some explanation based on the Internet topology and the AS hierarchy.

Approximate Reconstruction of Network Path Properties.

Once we have established an upper-bound to the monitoring cost this way, we develop NetQuest (Section 2.3) [90], a framework for *approximate reconstruction of network path properties* that provides far better scalability and flexibility with a slight sacrifice of accuracy. NetQuest is a flexible measurement framework that can support large-scale continuous network monitoring which consists of two key components: *design of experiments* and *network inference*.

We apply Bayesian experimental design to determine the set of active measurements that maximize the amount of information we gain about the network path properties subject to given resource constraints (*e.g.*, probing overhead). Bayesian experimental design is built on solid theoretical foundations, and has found numerous applications in scientific research and practical applications, ranging from software testing to medicine, to biology, and to car crash test. Recognizing its potential, we bring Bayesian experimental design into large-scale network measurement. Making the experimental design applicable to such context involves addressing several challenges. First, it is not clear how to formulate the problem of designing network measurement under the Bayesian experimental design framework. Second, the traditional Bayesian

experimental design often targets at a single application. In our environment, there can be many applications with different design requirements. How to use Bayesian experimental design to support such diverse application requirements is an interesting open problem.

To address the above issues, we first formulate the problem under the Bayesian experimental design framework. We then explore a series of Bayesian design schemes, and use extensive evaluation to identify the design scheme best suited for network monitoring. In addition, we develop techniques to achieve flexibility by designing measurement experiments that maximize the information gain for different design objectives and constraints. In particular, our approach can support the following requirements: (i) *differentiated design* for providing better resolution to certain parts of the network, (ii) *augmented design* for conducting additional measurements given existing observations, and (iii) *joint design* for supporting multiple users interested in different parts of the network.

Based on observations obtained from the measurements, we then use inference techniques to accurately reconstruct the global view of the network without requiring complete information. Our results show that our measurement framework can estimate network-wide average path delay within 15% error by monitoring within 2% paths. It achieves a similar degree of accuracy for estimating individual path properties by monitoring 10% paths. In addition, we demonstrate the flexibility of our measurement framework in providing differentiated monitoring, supporting continuous monitoring, and satisfying the

requirements of multiple users.

Contributions.

We make four main contributions. First, we provide a linear algebraic bound on the number of active probing that is required to exactly reconstruct path properties of networks. Through both synthetic and real overlay network topologies, we show that for reasonably large n (say 100), $k = O(n \log n)$ through linear regression tests on various synthetic and real topologies. We also provide some explanation based on the Internet topology and the AS hierarchy.

Second, our work brings Bayesian experimental design into large-scale network measurement. While Bayesian experimental design has found many applications in other scientific fields, to the best of our knowledge, this is the first time that it is applied to designing active network measurement experiments.

Third, building on top of Bayesian experimental design and inference techniques, we develop a unified framework within which a large class of network performance inference problems can be modeled, solved, and evaluated. Our framework is flexible, and can accommodate different design requirements. Our framework is also scalable, and can design measurement experiments that span thousands of routers and end hosts.

Fourth, we develop a toolkit that implements our framework on PlanetLab [80]. Using the toolkit we conduct an extensive evaluation of our frame-

work for efficient monitoring of end-to-end network performance. Our results demonstrate the effectiveness and flexibility of our framework.

2.1 Problem Formulation

Symbols	Meanings
P	set of all network paths
S	set of subset of P ($S \subseteq P$)
L	set of links on paths in P
m	number of end-to-end paths ($m = P $)
n	number of IP links ($n = L $)
$A \in \{0, 1\}^{m \times n}$	original routing matrix
$A_S \in \{0, 1\}^{S \times n}$	reduced routing matrix
$\mathbf{x} \in \mathbb{R}^n$	vector of unknown performance on individual links
$\mathbf{y} \in \mathbb{R}^m$	vector of observed end-to-end path performance
$\mathbf{y}_S \in \mathbb{R}^S$	reduced vector of observed path performance
v	vector in $\{0, 1\}^m$ (represents path)

Table 2.1: Table of notations

Formally, the problem can be specified as follows.

$$\mathbf{y} = A\mathbf{x}, \quad (2.1)$$

where \mathbf{x} is the vector of some unknown quantity, \mathbf{y} is the vector of observables, A is a matrix that associates \mathbf{y} and \mathbf{x} (often referred to as the *routing matrix*). In the context of network performance monitoring, \mathbf{x} is the vector of unknown performance on individual links, \mathbf{y} is the vector of observed performance on a set of end-to-end paths, and the routing matrix $A = (A_{ij})$ encodes whether link j belongs to path i , *i.e.*,

$$A_{ij} = \begin{cases} 1 & \text{if path } i \text{ contains link } j, \\ 0 & \text{otherwise.} \end{cases} \quad (2.2)$$

Note that our definition of routing matrix applies to both one-way and round-trip performance measurements. For round-trip measurements, the routing matrix can work for asymmetric routes.

The above formulation applies to any additive performance metric, such as delay or $\log\{1 - \text{loss rate}\}$. In the case of representing loss rate to additive metric, we first represent a path by $v \in \{0, 1\}^m$, where the j th entry v_j is one if link j is part of the path, and zero otherwise. Suppose link j drops packets with probability l_j ; then the loss rate p of a path represented by v is given by

$$1 - p = \prod_{j=1}^n (1 - l_j)^{v_j} \quad (2.3)$$

We take logarithms on both sides of (2.3). Then by \mathbf{x} with elements $\mathbf{x}_j = \log(1 - l_j)$, we can rewrite (2.3) as follows:

$$\log(1 - p) = \sum_{j=1}^n \mathbf{y}_j \log(1 - l_j) = \sum_{j=1}^n \mathbf{y}_j \mathbf{x}_j = \mathbf{y}^T \mathbf{x} \quad (2.4)$$

Let p_i be the end-to-end loss rate of the i th path, and let $\mathbf{y} \in \mathbb{R}^m$ be a column vector with elements $\mathbf{y}_i = \log(1 - p_i)$. Then we can obtain the m equations in form (2.4) as we did in 2.1. Therefore we can represent both delay and loss rate in the same form.

Besides performance estimation, there is another type of tomography problem, commonly referred to as *traffic matrix estimation*, which tries to infer end-to-end traffic demands based on observed link loads. In this context, \mathbf{x} is the vector of unknown traffic demands, \mathbf{y} is the vector of observed link loads.

There has been considerable recent progress on traffic matrix estimation [65, 109, 110]. In this work, we only consider network performance inference, but the framework and the techniques we develop can also be applied to traffic matrix estimation. We plan to explore this direction in our future research.

Our goal is to estimate $f(\mathbf{x})$, which is a function of link properties \mathbf{x} . One interesting example is $f(\mathbf{x}) = A\mathbf{x}$. In this case, the quantity of interest $f(\mathbf{x})$ represents properties of all network paths. More specifically, when \mathbf{x} is link delay, $f(\mathbf{x})$ is delay on all network paths. Another example is $f(\mathbf{x}) = \frac{1}{m}[1, 1, \dots, 1]_{1 \times m} A\mathbf{x}$, which corresponds to a network-wide average metric (*e.g.*, $f(\mathbf{x})$ is the network-wide average path delay, when \mathbf{x} is link delay).

In large-scale network measurement, it is too expensive to directly measure network properties on all paths. So the goal is to select only a small subset of the paths to probe so that we can still accurately estimate the quantity of interest. We formalize the path selection problem as follows.

Let P be the set of all network paths ($|P| = m$). Let L be the set of links appearing on paths in P ($|L| = n$). The performance on paths in P and the performance on links in L are related according to the linear system $\mathbf{y} = A\mathbf{x}$, where \mathbf{x} is a length- n column vector, \mathbf{y} is a length- m column vector, and A is a $m \times n$ routing matrix.

For any subset $S \subseteq P$ (with $s = |S|$), let A_S be the $s \times n$ sub-matrix of A formed by the s rows corresponding to those paths in S . Similarly, let

\mathbf{y}_S be the sub-vector of \mathbf{y} corresponding to the observed performance on those paths in S . The experimental design problem is to select a subset of paths S to probe such that we can estimate $f(\mathbf{x})$ based on the observed performance \mathbf{y}_S , A_S , and thus the following linear system

$$\mathbf{y}_S = A_S \mathbf{x}, \quad (2.5)$$

In this work, we consider the case when $f(\mathbf{x})$ is a linear function

$$f(\mathbf{x}) = F \mathbf{x}, \quad (2.6)$$

where F is a given $r \times n$ matrix.

The other major aspect of network performance monitoring is network inference. Its goal is to infer \mathbf{x} based on A_S and \mathbf{y}_S . The major challenge in network inference is that the linear system $\mathbf{y}_S = A_S \mathbf{x}$ is often under-determined due to partial observations, and can thus have an infinite number of solutions.

2.2 Exact Reconstruction of Network Path Properties

In this section, we illustrate the analysis technique for exact reconstruction of path properties. In particular, we address its algebraic model, basic static algorithms, scalability analysis, and evaluation.

2.2.1 Algebraic Model

In this section, we introduce an algebraic model of this work. We expand our notations as described in Table 2.2, in addition to Table 2.1.

Symbols	Meanings
N	number of end hosts on the overlay
t	number of identifiable links
$k \leq n$	rank of A
l_i	loss rate on i th link
p_i	loss rate on i th measurement path
\mathbf{x}_i	$\log(1 - l_i)$
\mathbf{y}_i	$\log(1 - p_i)$
p	loss rate along a path
$\mathcal{N}(A)$	null space of A
$\mathcal{R}(A^T)$	row(path) space of A ($= \text{range}(A^T)$)

Table 2.2: Additional table of notations

Suppose an overlay network spans n IP links. Normally, the number of paths m is much larger than the number of links n (see Fig. 2.1(a)). This suggests that we could select n paths to monitor, use those measurements to compute the link loss rate variables \mathbf{x} , and infer the loss rates of the other paths from (2.1).

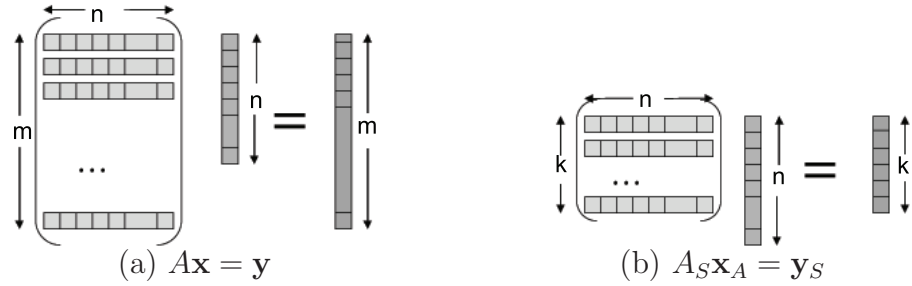


Figure 2.1: Matrix size representations.

However, as we discussed in Section 2.1, A is rank deficient: *i.e.*, $k = \text{rank}(A)$ and $k < n$. If A is rank deficient, we will be unable to determine the loss rate of some links from (2.1). These links are also called *unidentifiable* in

network tomography literature [11].

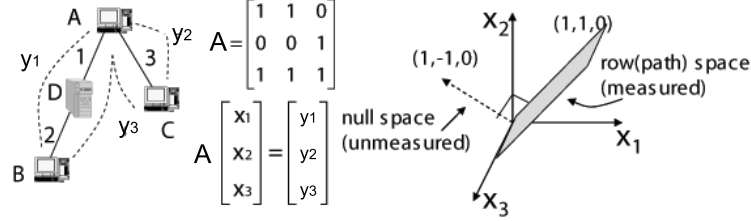


Figure 2.2: Sample overlay network.

Fig. 2.2 illustrates how rank deficiency can occur. There are three end hosts (A, B and C) on the overlay, three links (1, 2 and 3) and three paths between the end hosts. We cannot uniquely solve \mathbf{x}_1 and \mathbf{x}_2 because links 1 and 2 always appear together. We know their sum, but not their difference.

Fig. 2.2 illustrates the geometry of the linear system, with each variable \mathbf{x}_i as a dimension. The vectors $\{\alpha [1 \ -1 \ 0]^T\}$ comprise $\mathcal{N}(A)$, the *null space* of A . No information about the loss rates for these vectors is given by (2.1). Meanwhile, there is an orthogonal *row(path) space* of A , $\mathcal{R}(A^T)$, which for this example is a plane $\{\alpha [1 \ 1 \ 0]^T + \beta [0 \ 0 \ 1]^T\}$. Unlike the null space, the loss rate of any vector on the row space can be uniquely determined by (2.1).

To separate the identifiable and unidentifiable components of \mathbf{x} , we decompose \mathbf{x} into $\mathbf{x} = \mathbf{x}_A + \mathbf{x}_N$, where $\mathbf{x}_A \in \mathcal{R}(A^T)$ is its projection on the row space and $\mathbf{x}_N \in \mathcal{N}(A)$ is its projection on the null space (*i.e.*, $A\mathbf{x}_N = 0$). The decomposition of $[\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3]^T$ for the sample overlay is shown below.

$$\mathbf{x}_A = \frac{(\mathbf{x}_1 + \mathbf{x}_2)}{2} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} + \mathbf{x}_3 \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1/2 \\ \mathbf{y}_1/2 \\ \mathbf{y}_2 \end{bmatrix} \quad (2.7)$$

$$\mathbf{x}_N = \frac{(\mathbf{x}_1 - \mathbf{x}_2)}{2} \begin{bmatrix} 1 \\ -1 \\ 0 \end{bmatrix} \quad (2.8)$$

Thus the vector \mathbf{x}_A can be uniquely identified, and contains all the information we can know from (2.1) and the path measurements. The intuition of our scheme is illustrated through *virtual links* in [15].

Because \mathbf{x}_A lies in the k -dimensional space $\mathcal{R}(A^T)$, only k independent equations of the m equations in (2.1) are needed to uniquely identify \mathbf{x}_A . We measure these k paths to compute \mathbf{x}_A . Since $\mathbf{y} = A\mathbf{x} = A\mathbf{x}_A + A\mathbf{x}_N = A\mathbf{x}_A$, we can compute all elements of \mathbf{y} from \mathbf{x}_A , and thus obtain the loss rate of all other paths. Next, we present more detailed algorithms.

2.2.2 Basic Static Algorithms

The basic algorithms involve two steps. First, we select a basis set of k paths to monitor. Such selection only needs to be done once at setup. Then, based on continuous monitoring of the selected paths, we calculate and update the loss rates of all other paths.

2.2.2.1 Measurement Paths Selection

To select k linearly independent paths from A , we use standard rank-revealing decomposition techniques [38], and obtain the reduced system (Eq. 2.5) discussed in Section 2.1. where $A_S \in \mathbb{R}^{k \times n}$ and $\mathbf{y}_S \in \mathbb{R}^k$ consist of k rows of A and \mathbf{y} , respectively. The equation is illustrated in Fig. 2.1(b) (compared with $A\mathbf{x} = \mathbf{y}$).

As shown below, our algorithm is a variant of the QR decomposition with column pivoting [38, p.223]. It incrementally builds a decomposition $A_S^T = QR$, where $Q \in \mathbb{R}^{n \times k}$ is a matrix with orthonormal columns and $R \in \mathbb{R}^{k \times k}$ is upper triangular.

```

procedure SelectPath(A)
1 for every row(path)  $v$  in  $A$  do
2    $\hat{R}_{12} = R^{-T} A_S v^T = Q^T v^T$ 
3    $\hat{R}_{22} = \|v\|^2 - \|\hat{R}_{12}\|^2$ 
4   if  $\hat{R}_{22} \neq 0$  then
5     Select  $v$  as a measurement path
6     Update  $R = \begin{bmatrix} R & \hat{R}_{12} \\ 0 & \hat{R}_{22} \end{bmatrix}$  and  $A_S = \begin{bmatrix} A_S \\ v \end{bmatrix}$ 
   end if
end for

```

Figure 2.3: Path (row) selection algorithm

In general, the A matrix is very sparse; that is, there are only a few nonzeros per row. We leverage this property for speedup. We further use optimized routines from the LAPACK library [6] to implement row selection algorithm so that it inspects several rows at a time. The complexity of row selection algorithm is $O(mk^2)$, and the constant in the bound is modest. The memory cost is roughly $k^2/2$ single-precision floating point numbers for storing the R factor. Notice that the path selection only needs to be executed once for initial setup.

2.2.2.2 Path Loss Rate Calculations

To compute the path loss rates, we must find a solution to the underdetermined linear system $A_S \mathbf{x}_A = \mathbf{y}_S$. The vector \mathbf{y}_S comes from measurements

of the paths. Zhang *et al.* report that path loss rates remain operationally stable in the time scale of an hour [107], so these measurements need not be taken simultaneously.

Given measured values for \mathbf{y}_S , we compute a solution \mathbf{x}_A using the QR decomposition we constructed during measurement path selection [38, 26]. We choose the unique solution \mathbf{x}_A with minimum possible norm by imposing the constraint $\mathbf{x}_A = A_S^T \mathbf{z}$ where $\mathbf{z} = R^{-1} R^{-T} \mathbf{y}_S$. Once we have \mathbf{x}_A , we can compute $\mathbf{y} = A \mathbf{x}_A$, and from there infer the loss rates of the unmeasured paths. The complexity for this step is only $O(k^2)$. Thus we can update loss rate estimates online.

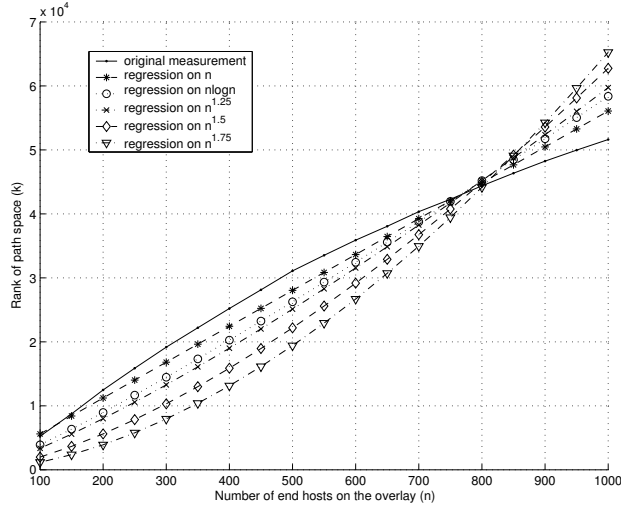


Figure 2.4: Regression of k in various functions of n under different router-level topologies.

2.2.3 Scalability Analysis

An overlay monitoring system is scalable only when the size of the basis set, k , grows relatively slowly as a function of n . Given that the Internet has moderate hierarchical structure [98, 34], we proved that the number of end hosts is no less than half of the total number of nodes in the Internet. Furthermore, we proved that when all the end hosts are on the overlay network, $k = O(N)$ [15].

But what about if only a small fraction of the end hosts are on the overlay? Because A is an m by n matrix, k is bounded by the number of links n . If the Internet topology is a strict hierarchy like a tree, $n = O(N)$, thus $k = O(N)$. But if there is no hierarchy at all (*e.g.* a clique), $k = O(N^2)$ because all the $O(N^2)$ paths are linearly independent. Tangmunarunkit *et al.* found that the power-law degree Internet topology has moderate hierarchy [98]. It is our conjecture that $k = O(N \log n)$.

In this section, we will show through linear regression on real topologies that k is indeed bounded by $O(N \log N)$ for reasonably large N (*e.g.* 100). We explain it based on the power-law degree distribution of the Internet topology and the AS (Autonomous System) hierarchy.

We randomly select end hosts which have the least degree (*i.e.*, leaf nodes) to form an overlay network. We test by linear regression of k on $O(N)$, $O(N \log N)$, $O(N^{1.25})$, $O(N^{1.5})$, and $O(N^{1.75})$. As shown in Fig. 2.4, results are averaged over three runs with different random sets of end hosts. We

find that $O(N)$ regression has the least residual errors - actually k even grows slower than $O(N)$. Considering the simulated topological models not included in this dissertation, we can conservatively say $k = O(N \log N)$.

Note that such trend still holds when the end hosts are sparsely distributed in the Internet, *e.g.*, when each end host is in a different access network. One extreme case is the “star” topology - each end host is connected to the same center router via its own access network. In such a topology, there are only N links. Thus $k = O(N)$. Only topologies with very dense connectivity, like a full clique, have $k = O(N^2)$. Those topologies have little link sharing among the end-to-end paths.

The key observation is that when N is sufficiently large, such dense connectivity is very unlikely to exist in the Internet because of the power-law degree distribution. Tangmunarunkit *et al.* found that link usage, as measured by the set of node pairs (source-destination pairs) whose traffic traverses the link, also follows a power-law distribution, *i.e.*, there is a very small number of links that are on the shortest paths of the majority of node pairs. So there is significant amount of link sharing among the paths, especially for backbone links, customer links, and peering links.

Such link sharing can easily lead to rank deficiency of the path matrix for overlay networks. As an example, consider an overlay within a single AS. The AS with the largest number of links (exclusive of customer and peering links) in [92] has 5,300 links. Even considering the coverage factor (55.6% as in Table 2 of [92]), there are at most 9,600 links. Since there are $N(N - 1)$ paths

among n nodes, link sharing must occur before $N = 100$; in fact, substantial link sharing is likely to occur for even smaller N .

Now consider an overlay network that spans two ASes connected by c customer/peering links, with $N/2$ nodes in one AS and $N/2$ nodes in the other. The $N^2/2$ cross-AS paths can be modelled as linear combination of $2c \times N + 2c$ virtual links - bi-directional links from each end host to its c peering link routers, and c bi-directional peering links. Thus given c is normally much less than N and can be viewed as a constant, only $O(N)$ paths need to be measured for the $O(N^2)$ cross-AS paths.

Now consider an overlay on multiple ASes. According to [96], there are only 20 ASes (*tier-1* providers) which form the dense core of the Internet. These ASes are connected almost as a clique, while the rest of the ASes have far less dense peering connectivity. So when the size of an overlay is reasonably big (*e.g.*, $N > 100$), the number of customer and peering links that cross-AS paths traverse tends to grow much slower than $O(N^2)$. For example, a joining end host may only add one customer link to the overlay topology, and share the peering links that have been used by other end hosts. Meanwhile, only a few nodes are needed in a single AS before link sharing occurs in paths within an AS.

We believe this heavy sharing accounts for our empirical observation that $k = O(N)$ in a real router-level topology, and k grows at worst like $O(N \log N)$ in several generated topologies. Note that the real 284,805-router topology represents a fairly complete transit portion of the Internet [42]. In our

analysis, we conservatively assume that there is only one end host connecting to each edge router to reduce the possible path sharing, but we still find $k = O(N)$ when $N > 100$.

2.3 Approximate Reconstruction of Network Path Properties

In this section, we illustrate a network measurement framework named NetQuest which approximately reconstructs path properties. In the subsequent subsections, we describe design of experiments, inference algorithms, and evaluation of NetQuest.

2.3.1 Design of Experiments

Network measurement, especially when conducted at large scale, requires carefully designed measurement experiments. The design involves specifying all aspects of an experiment and choosing the values of variables that can be controlled before the experiment starts. Making the design decisions is challenging given the complexity involved in experiments. Manually making all the design decisions is both time consuming and error prone. It is therefore highly desirable to automate the design process and do so in a mathematically sound manner. Not all aspects of experimental design are amenable to formal mathematical treatment. Choosing the values for the control variables however can be expressed in a coherent mathematical framework through the use of *Bayesian experimental design*, which has gained considerable popularity in the past three decades. Below we first give a brief overview of Bayesian experimental design (see [13, 20] for a detailed review). We then put it into the context of large-scale network measurement and demonstrate how the general framework can be applied to meet common design requirements.

2.3.1.1 Bayesian Experimental Design

The basic idea in experimental design is that one can improve the statistical inference about the quantities of interest by properly choosing the values of the control variables. This can be formally described in a Bayesian decision theoretic framework as proposed by Lindley in 1972 [60, page 19 and 20]. Below we first set up the framework using decision theoretic terminologies, and then put it into the context of network performance monitoring.

As summarized in [13], Lindley’s framework is the following. Suppose one wants to conduct an experiment on a system with unknown parameters \mathbf{x} drawn from parameter space \mathcal{X} . Before the experiment, a design η must be chosen from some set \mathcal{H} . Through the experiment, data \mathbf{y} from a sample space \mathcal{Y} will be observed. Based on \mathbf{y} a terminal decision d will be chosen from some set \mathcal{D} . In the context of network performance monitoring, the terminal decision d is an estimate of our quantity of interest $f(\mathbf{x})$, where \mathbf{x} is the vector of unknown link performance. A design η refers to a set of paths, S , which we choose to probe. Through the experiment, we observe end-to-end performance on the set of paths in S : \mathbf{y}_S , which satisfies $\mathbf{y}_S = A_S \mathbf{x}$. Here A_S is the routing matrix formed by the set of rows corresponding to paths in S . So the goal is to estimate $f(\mathbf{x})$ based on the observed end-to-end performance on the set of paths in S (*i.e.*, \mathbf{y}_S). So the whole decision process consists of two parts: first the selection of η (*i.e.*, S , in our context), and then the choice of a terminal decision d (*i.e.*, an estimate of $f(\mathbf{x})$, in our context). A general utility function of the form $U(d, \mathbf{x}, \eta, \mathbf{y})$ is used to reflect the purpose of the experiment. For

example, it may reflect the expected accuracy of an estimator for $f(\mathbf{x})$ in the context of network performance inference.

Bayesian experimental design suggests that a good solution to the experimental design problem is the design that maximizes the expected utility of the best terminal decision. More formally, for a given design η , the expected utility of the best terminal decision is

$$U(\eta) = \int_{\mathbf{y}} \max_{d \in D} \int_{\mathbf{x}} U(d, \mathbf{x}, \eta, \mathbf{y}) p(\mathbf{x}|\mathbf{y}, \eta) p(\mathbf{y}|\eta) d\mathbf{x} d\mathbf{y}, \quad (2.9)$$

where $p(\cdot)$ denotes a probability density function with respect to an appropriate measure. Bayesian experimental design would then choose the design η^* that maximizes the above $U(\eta)$:

$$U(\eta^*) = \max_{\eta \in \mathcal{H}} \int_{\mathbf{y}} \max_{d \in D} \int_{\mathbf{x}} U(d, \mathbf{x}, \eta, \mathbf{y}) p(\mathbf{x}|\mathbf{y}, \eta) p(\mathbf{y}|\eta) d\mathbf{x} d\mathbf{y}. \quad (2.10)$$

1) Bayesian Designs for Path Selection

We now apply Bayesian experimental design to solve the path selection problem. Different Bayesian designs can be obtained by choosing different utility functions to assess the quality of individual designs. Below we introduce two such designs: *Bayesian A-optimal design* and *Bayesian D-optimal design*.

Bayesian A-optimal design: For estimating $f(\mathbf{x}) = F\mathbf{x}$, we can use the squared error $\|F\mathbf{x} - F\mathbf{x}_e\|_2^2 = (F\mathbf{x} - F\mathbf{x}_e)^T (F\mathbf{x} - F\mathbf{x}_e)$ to assess the inaccuracy of an estimator $F\mathbf{x}_e$. So a design η can be chosen to maximize the

following expected utility

$$U_A(\eta) = - \int (F\mathbf{x} - F\hat{\mathbf{x}})^T (F\mathbf{x} - F\hat{\mathbf{x}}) p(\mathbf{y}, \mathbf{x}|\eta) d\mathbf{x} d\mathbf{y}, \quad (2.11)$$

where $\hat{\mathbf{x}}$ is the estimated \mathbf{x} under the best decision rule d .

To derive an easy-to-use design criterion, we will assume a normal linear system. Specifically, we assume that $\mathbf{y}_S|\mathbf{x}, \sigma^2 \sim A_S\mathbf{x} + N(0, \sigma^2 I)$, where σ^2 is the known variance for the zero mean Gaussian measurement noise, and I is the identity matrix. Suppose the prior information is that $\mathbf{x}|\sigma^2$ is randomly drawn from a multivariate normal distribution with mean vector μ and covariance matrix $\Sigma = \sigma^2 R^{-1}$, where μ and matrix R are known *a priori*.

Let $D(\eta) = (A_S^T A_S + R)^{-1}$. The Bayesian procedure yields

$$U_A(\eta) = -\sigma^2 \text{tr}\{FD(\eta)F^T\}, \quad (2.12)$$

where $\text{tr}\{M\}$ (the trace of a matrix M) is defined as the sum of all the diagonal elements of M .

Maximizing $U_A(\eta)$ thus reduces to minimizing the function

$$\phi_A(\eta) = \text{tr}\{FD(\eta)F^T\}, \quad (2.13)$$

which is commonly referred to as the *Bayesian A-optimality*.

Bayesian D-optimal design: It is also common to replace the quadratic error function in Bayesian *A-optimality* with an information-theoretic metric. Specifically, one can choose a design that maximizes the expected gain

in Shannon information or, equivalently, maximizes the expected Kullback-Leibler distance between the posterior and the prior distributions:

$$\int \log \frac{p(F\mathbf{x}|\mathbf{y}, \eta)}{p(F\mathbf{x})} p(\mathbf{y}, \mathbf{x}|\eta) d\mathbf{x} d\mathbf{y}. \quad (2.14)$$

Since the prior distribution of $F\mathbf{x}$ does not depend on the design η , maximizing (2.14) is equivalent to maximizing

$$U_D(\eta) = \int \log \{p(F\mathbf{x}|\mathbf{y}, \eta)\} p(\mathbf{y}, \mathbf{x}|\eta) d\mathbf{x} d\mathbf{y}. \quad (2.15)$$

Under a normal linear model, the Bayesian procedure yields

$$U_D(\eta) = -\frac{n}{2} \log(2\pi) - \frac{n}{2} - \frac{1}{2} \log \det\{FD(\eta)F^T\}, \quad (2.16)$$

where $\det\{M\}$ denotes the determinant of a matrix M .

Maximizing $U_D(\eta)$ thus reduces to minimizing the function

$$\phi_D(\eta) = \det\{FD(\eta)F^T\}, \quad (2.17)$$

which we define as the *Bayesian D-optimality*.

One problem with (2.17) is that when $\text{rank}(F) < r$ (r is the number of rows in F), $\det\{FD(\eta)F^T\}$ is always 0 and thus cannot distinguish different designs. Our solution is to redefine

$$\phi_D(\eta) = \Pi_{\text{rank}(F)}\{FD(\eta)F^T\}, \quad (2.18)$$

where $\Pi_k\{M\}$ is the product of the k largest eigenvalues of M .

It is often reasonable to require that a design η remains good under a small perturbation to the function of interest: $f(\mathbf{x}) = F\mathbf{x}$. As a result, if F is not full-rank, we can perturb F slightly to make it full-rank. Therefore, we only need to consider the case when $\text{rank}(F) = \min(r, n)$. In this case, we can simplify (2.18) into

$$\phi_D(\eta) = \begin{cases} \det\{F^T F\} \det\{D(\eta)\} & \text{if } r \geq n, \\ \det\{FD(\eta)F^T\} & \text{otherwise.} \end{cases} \quad (2.19)$$

Note that when $r \geq n$, minimizing $\phi_D(\eta)$ reduces to minimizing $\det\{D(\eta)\}$ or, equivalently, maximizing $\det\{A_S^T A_S + R\}$.

2) Search Algorithm

Once we have chosen a design criterion $\phi(\eta)$, the next step is to find the optimal design $\eta^* = \arg \min_{\eta} \phi(\eta)$. However, the problem of finding s rows of A to minimize a given design criterion $\phi(\eta)$ is known to be *NP*-complete and it is computationally infeasible to compute the optimal design exactly. To address the issue, we search for a good design using a simple *sequential search algorithm*. The algorithm starts with an empty initial design and then sequentially adds rows to the design. During each iteration, it greedily selects the row that results in the largest reduction in $\phi(\eta)$. The basic algorithm is illustrated in Figure 2.5.

It is possible to further improve the design obtained by the sequential search algorithm using an *exchange algorithm* (e.g., [35, 69, 66, 76]), which tries to reduce $\phi(\eta)$ by iteratively exchanging paths. We have implemented

```

1   $S = \emptyset$  // initialize the path set to be empty
2  for  $iter = 1$  to  $s$  //  $s$  is the desired number of paths
3      for each path  $i \in \{1, 2, \dots, m\} - S$ 
4          // compute the new design criterion after adding path  $i$ 
5           $criteria[i] = \phi(S \cup \{i\})$ 
6      end for
7      // select the path that minimizes the new design criterion
8       $S = S \cup \{\arg \min_i criteria[i]\}$ 
9  end for
10 return  $S$ 

```

Figure 2.5: Sequential path selection for Bayesian designs.

Fedorov’s exchange algorithm, which has been shown to yield the best performance among many existing algorithms [76]. However, our experience suggests that the additional path exchanges do not yield noticeable performance improvement in the context of network performance inference. *So we disable the exchange algorithm in the interest of efficiency.*

3) Incremental Update of Design Criterion

For large-scale network measurement, it is essential for the search algorithm to be highly efficient, because the design space is often very large due to the quadratic growth in the number of network paths with respect to the number of network nodes. The major bottleneck of the above search algorithm is computing the new design criterion $\phi(S \cup \{i\})$ after adding a path i (line 5 in Figure 2.5). Recall that both $\phi_A(\eta)$ and $\phi_D(\eta)$ are functions of $FD(\eta)F^T = F(A_S^T A_S + R)^{-1}F^T$. Given the size of $(A_S^T A_S + R)$ and F , it is inefficient to compute $\phi(S \cup \{i\})$ from scratch due to the expensive matrix inversion and matrix multiplication operations involved.

In NetQuest, we significantly improve the efficiency of the search algorithm by applying incremental methods to update the design criterion. With such optimizations, NetQuest has successfully handled routing matrices A with 1,000,000 rows, 50,000 columns, and over 5,000,000 non-zero entries (corresponding to paths among 1,000 nodes). Below we present these incremental update methods in detail.

Notations: Let $S' = S \cup \{i\}$, $M = A_S^T A_S + R$, $M' = A_{S'}^T A_{S'} + R$, $N = FM^{-1}F^T$, $N' = F(M')^{-1}F^T$. With these notations, we have $\phi_A(S) = \text{tr}\{N\}$, $\phi_A(S') = \text{tr}\{N'\}$, $\phi_D(S) = \det\{N\}$, $\phi_D(S') = \det\{N'\}$.

Incremental computation of $\phi_A(S \cup \{i\})$: Let \mathbf{a}_i^T denote the i -th row vector of A . It is easy to verify that $M' = M + \mathbf{a}_i \mathbf{a}_i^T$. That is, M' can be derived from M using a rank-1 matrix update. We have the following result from matrix algebra

$$(M')^{-1} = (M + \mathbf{a}_i \mathbf{a}_i^T)^{-1} = M^{-1} - \alpha \mathbf{u} \mathbf{u}^T, \quad (2.20)$$

where $\mathbf{u} = M^{-1} \mathbf{a}_i$, $\alpha = 1/(1 + \mathbf{a}_i^T \mathbf{u})$.

Combine (2.20) with $N = FM^{-1}F^T$ and $N' = F(M')^{-1}F^T$, we obtain $N' = N - \alpha(F\mathbf{u})(F\mathbf{u})^T$. As a result, we have

$$\text{tr}\{N'\} = \text{tr}\{N\} - \text{tr}\{\alpha(F\mathbf{u})(F\mathbf{u})^T\} = \text{tr}\{N\} - \alpha \|F\mathbf{u}\|_2^2. \quad (2.21)$$

Therefore, we can compute $\phi_A(S \cup \{i\}) = \text{tr}\{N'\}$ incrementally by computing $\mathbf{u} = M^{-1} \mathbf{a}_i$, $\alpha = 1/(1 + \mathbf{a}_i^T \mathbf{u})$, and $\|F\mathbf{u}\|_2^2$ (note that M^{-1} remains

fixed for different i). These operations are much more efficient than matrix inversion and matrix multiplication, which are required to compute $\phi_A(S \cup \{i\})$ from scratch.

Incremental computation of $\phi_D(S \cup \{i\})$: Let $\mathbf{b} = \sqrt{\alpha} \cdot F\mathbf{u}$. We have $N' = N - \mathbf{b}\mathbf{b}^T$. Using results in matrix algebra, we have

$$(N')^{-1} = (N - \mathbf{b}\mathbf{b}^T)^{-1} = N^{-1} + \beta \mathbf{v}\mathbf{v}^T, \quad (2.22)$$

$$\det\{N'\} = \det\{N - \mathbf{b}\mathbf{b}^T\} = \frac{1}{\beta} \det\{N\}, \quad (2.23)$$

where $\mathbf{v} = N^{-1}\mathbf{b}$, and $\beta = 1/(1 - \mathbf{b}^T\mathbf{v})$.

Therefore, we can compute $\phi_D(S \cup \{i\}) = \det\{N'\}$ incrementally by computing $\mathbf{v} = N^{-1}\mathbf{b}$, and $\beta = 1/(1 - \mathbf{b}^T\mathbf{v})$ (note that N^{-1} remains fixed for different i). These operations are much more efficient than the matrix inversion and matrix multiplication operations required for computing $\phi_D(S \cup \{i\})$ from scratch.

Further optimization: With the above incremental update methods, we need to update M^{-1} and N^{-1} each time after a new path is added to S (line 8 in Figure 2.5). This takes $O(n^2)$ operations using (2.20) and (2.22). We can further improve efficiency by maintaining the Cholesky factorization of M and N (instead of M^{-1} and N^{-1}), which in general are much sparser and thus more efficient to update incrementally. Note that the only use of M^{-1} and N^{-1} is to compute quantities $\mathbf{u} = M^{-1}\mathbf{a}_i$ and $\mathbf{v} = N^{-1}\mathbf{b}$. We can compute the same quantities using the Cholesky factorization. For example, if we

write $M = LL^T$, where L is the lower-triangular factorization of M , we have $\mathbf{u} = (L^T)^{-1}(L^{-1}\mathbf{a}_i)$, which can be computed efficiently using back-substitution without inverting L and L^T .

In NetQuest, we use LDL [25], a MATLAB package highly optimized for incremental Cholesky factorization of sparse matrices. Our experience suggests that the resulting algorithm achieves an order of magnitude speedup over directly updating M^{-1} and N^{-1} .

2.3.1.2 Flexibility

Our Bayesian experimental design framework is very flexible and can accommodate common requirements in large-scale network measurement. Below we cover three common scenarios.

Differentiated design: In large-scale network performance monitoring, different quantities of interest may not always have the same importance. For example, a subset of paths may belong to a major customer and it is important to monitor those paths more extensively than the other paths. We can accommodate such differentiated design requirement in Bayesian A -optimality by assigning higher weights to those important rows of matrix F in the objective function $f(\mathbf{x}) = F\mathbf{x}$.

Augmented design: Augmented design is useful in large-scale network measurement for several reasons. First, when some of the measurements in a previous design failed, we do not want to design a new experiment completely from scratch, but instead would like to leverage the data that we have already

observed as much as possible. Second, augmented design can also be used to design active measurement experiments that complement well with the existing passive measurement (*i.e.*, the additional information gain is maximized). Our design framework can naturally support the augmentation of a previous design. Specifically, let S_0 be the set of rows we obtain in the previous design. We just need to redefine

$$D(\eta) = (A_{S \cup S_0}^T A_{S \cup S_0} + R)^{-1} = (A_S^T A_S + R + A_{S_0}^T A_{S_0})^{-1} \quad (2.24)$$

where $S \cap S_0 = \emptyset$. We can then apply the Bayesian A -optimal and D -optimal design criteria to find S , the set of additional paths to probe. In addition, we also extend QR and SVD, which will be described in Section 2.3.1.3, to support augmented design by excluding the paths with successful measurements and applying SVD or QR to select a set of paths from the remaining rows.

Multi-user design: In large-scale network performance monitoring, there may be multiple users who are interested in different parts of the network and may have different functions of interest. We can support such scenarios by using a linear combination of individual users' design criteria as the overall design criterion. Formally, given a set of users $1, 2, \dots, u$, let $\phi_i(\eta)$ be the preferred design criterion for each user i (which may depend on his/her interest: $f_i(\mathbf{x}) = F_i \mathbf{x}$). We can then use $\phi(\eta) = \sum_i w_i \phi_i(\eta)$ as the combined design criterion, where w_i is the weight associated with user i . As a concrete example, consider the case where Bayesian A -optimality is used by all the users. In this

case, we have $\phi_i(\eta) = \text{tr}\{F_i D(\eta) F_i^T\}$. We can show

$$\phi(\eta) = \sum_i w_i \text{tr}\{F_i D(\eta) F_i^T\} = \text{tr}\{F D(\eta) F^T\}, \quad (2.25)$$

where $F = \text{vertcat}\{w_i^{1/2} F_i\}$ is the vertical concatenation of matrices $w_i^{1/2} F_i$. Therefore, the optimal design using the combined design criterion is equivalent to the Bayesian A -optimal design for the combined function: $f(\mathbf{x}) = F\mathbf{x}$. Note that if a subset of users (U) share a common row in their F_i , this row will appear as multiple rows in F . These rows can be merged into a single row with a combined weight of $(\sum_{i \in U} w_i)^{1/2}$. In the special case when $w_i = 1$, the combined weight is simply $|U|^{1/2}$, *i.e.*, the square root of the number of users interested in the row.

Besides the above three common scenarios, our design framework can easily incorporate other constraints in the design space (*e.g.*, the maximum amount of paths that one can probe at each monitoring site, and the number of monitoring sites available). As part of our future work, we plan to further investigate them in the context of specific network monitoring applications.

2.3.1.3 Non-Bayesian Designs

For performance comparison, we will also examine the following non-Bayesian solutions to the path selection problem.

Rank based solution: In Section 2.2, we presented a linear algebraic approach to efficient monitoring of overlay paths. In the context, given N overlay nodes, there are $N(N - 1)$ different paths. So A has $N(N - 1)$ rows. The

quantity of interest is $f(\mathbf{x}) = A\mathbf{x}$ (*i.e.*, the performance on all overlay paths). Given the rank of matrix A , k , the solution is based on the observation that any subset of k independent rows of A , denoted as A_S , are sufficient to span the space of A : $\{\mathbf{y} \in \mathbb{R}^m | \exists \mathbf{x} \in \mathbb{R}^n \text{ s.t. } \mathbf{y} = A\mathbf{x}\}$. As a result, given the measurements for paths corresponding to the rows in A_S , one can reconstruct the end-to-end performance on all paths *exactly*. As we have seen from Section 2.2.3, k gives an upper bound on the number of paths that one needs to probe.

SVD based solution: Chua *et al.* [18, 19] presented a statistical framework for monitoring a linear summary of the end-to-end path performance. Their quantity of interest is $f(\mathbf{x}) = \ell^T A\mathbf{x}$, where ℓ is a weight vector. This is a network-wide metric, *e.g.*, average delay of all paths in a network. It is a special case of the inference problem we study in this work.

Chua *et al.* observed that routing matrices encountered in practice generally show significant sharing of links between paths. As a result, A tends to have small *effective* rank compared to their actual matrix rank. That is, a small subset of eigenvalues of $A^T A$ tend to be much larger than the rest. Based on the observation, Chua *et al.* proposed to select a subset of s paths such that the corresponding rows span as much of $\mathcal{R}(A)$ as possible, where $\mathcal{R}(A)$ is the subspace formed by all possible linear combinations of the rows in A . Algorithmically, this problem is equivalent to the subset selection problem in the field of computational linear algebra (see [40, Ch 12]). So [18] adapted the method described in Algorithm 12.2.1 of [40, p. 574], which is based on the

- 1 compute C such that $\Sigma = CC^T$
- 2 compute SVD of AC : $U\nabla V^T = AC$
- 3 extract the first s column vectors of U : $U_s = [\mathbf{u}_1 \mathbf{u}_2 \cdots \mathbf{u}_s]$
- 4 compute the QR factorization with column pivoting on U_s^T :
 $QR = (U_s[E, \cdot])^T$ (E is a permutation vector for rows of U_s)
- 5 return the first s elements of E : $S = \{e_1, e_2, \cdots, e_s\}$

Figure 2.6: SVD based path selection algorithm

singular value decomposition (SVD). Subsequently in [19], Chua *et al.* extends their algorithm to incorporate Σ , the covariance matrix of \mathbf{x} . It assumes that Σ is a diagonal matrix (but allows diagonal elements to be different). The resulting algorithm is summarized in Figure 2.6.

Path selection under general link covariance Σ (*e.g.*, when link performance has spatial dependency) is left as an *open* problem in [19]. Our Bayesian experimental design framework works for any link covariance matrix, and therefore solves this problem.

QR based solution: The third alternative solution is directly based on QR factorization with column pivoting. It is one of the two algorithms proposed by Golub *et al.* [39] for selecting *numerically* independent rows/columns (the other algorithm is the SVD based solution described above). As noted in [39], the QR based solution is generally more efficient than the SVD based solution and often achieves comparable performance. We extend the algorithm in [39] to incorporate the link covariance matrix Σ when Σ is a diagonal matrix. The resulting algorithm is illustrated in Figure 2.7.

Summary: Rank based solution requires monitoring $\text{rank}(A)$ number of

- | | |
|---|---|
| 1 | compute C such that $\Sigma = CC^T$ |
| 2 | compute $G = (AC)^T$ |
| 3 | compute the QR factorization with column pivoting on G :
$QR = G[\cdot, E]$ (E is a permutation vector for columns of G) |
| 4 | return the first s elements of E : $S = \{e_1, e_2, \dots, e_s\}$ |

Figure 2.7: QR based path selection algorithm

paths, which can be expensive in large networks. SVD and QR based solutions can monitor fewer paths at the cost of higher error. We further enhance the flexibility of SVD and QR by extending them to support augmented design. Nevertheless, as we will show, Bayesian experimental design can out-perform the alternatives in the following regards: (i) it achieves higher accuracy when monitoring the same number of paths as SVD and QR, (ii) it is scalable and can be applied to networks of thousands of nodes, and (iii) it is flexible to support diverse design requirements.

2.3.2 Inference Algorithms

The other major component to the NetQuest framework is network inference, which reconstructs the quantities of interest \mathbf{x} based on partial, indirect observations \mathbf{y} by solving (2.1). Since NetQuest selects only a small number of measurement experiments to conduct, the number of observables can be much smaller than the number of unknowns. Therefore, the linear inverse problem in (2.1) is often *under-determined*. A lot of solutions have been developed for under-determined linear inverse problems. As we noted in

Notation	Inference algorithm	Section
MinL2	L_2 norm minimization	§2.3.2.1
MinL1	L_1 norm minimization	§2.3.2.2
Entropy	Maximum Entropy Estimation	§2.3.2.3

Table 2.3: Inference algorithms

[110], many such proposals solve the regularized least-squares problem

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 J(\mathbf{x}), \quad (2.26)$$

where $\|\cdot\|_2$ denotes the L_2 norm, λ is a regularization parameter, and $J(\mathbf{x})$ is a penalization functional. Proposals of this kind have been used in a wide range of fields, with considerable practical and theoretical success when the data match the assumptions of the method, and the regularization functional matches the properties of the estimand. See [44] for a general description of regularization.

In this work, we compare the accuracy of several widely used inference algorithms as summarized in Table 2.3. **MinL2** and **MinL1** can optionally incorporate the nonnegativity constraints: $\mathbf{x} \geq 0$, resulting in **MinL2_nonNeg** and **MinL1_nonNeg**, respectively. The goal is to understand how combinations of different experimental design methods and inference algorithms affect the overall inference accuracy.

2.3.2.1 L_2 Norm Minimization

A common solution to (2.1) is L_2 norm minimization, which corresponds to (2.26) with $J(\mathbf{x}) = \|\mathbf{x}\|_2^2$.

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \|\mathbf{x}\|_2^2. \quad (2.27)$$

If we have prior information that \mathbf{x} is close to an initial solution μ , we can replace $\|\mathbf{x}\|_2$ with $\|\mathbf{x} - \mu\|_2$ in (2.27), resulting in

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \|\mathbf{x} - \mu\|_2^2. \quad (2.28)$$

(2.28) is also commonly referred to as the Tikhonov regularization [44]. It can be efficiently solved using standard solvers for linear least-squares problems. If desired, one can incorporate the nonnegativity constraints $\mathbf{x} \geq 0$ into (2.28). The resulting nonnegative linear least-squares problem can be solved using PDCO [85], a MATLAB package highly optimized for problems with sparse matrices A .

2.3.2.2 L_1 Norm Minimization

Another common solution to (2.1) is L_1 norm minimization, which corresponds to (2.26) with $J(\mathbf{x}) = \|\mathbf{x}\|_1$ (*i.e.*, the L_1 norm of \mathbf{x}).

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \|\mathbf{x}\|_1. \quad (2.29)$$

L_1 norm minimization is often used in situations where \mathbf{x} is *sparse*, *i.e.*, \mathbf{x} has only very few large elements and the other elements are all close to 0.

This can happen, for instance, in loss inference, where most links have close to 0 loss rate (and thus $\log\{1 - \text{loss rate}\}$ is close to 0). In such scenarios, ideally one would like to find the sparsest solution to $\mathbf{y} = A\mathbf{x}$ by minimizing the L_0 norm $\|\mathbf{x}\|_0$ (*i.e.*, the number of nonzeros in \mathbf{x}). But since the L_0 norm is not convex and is notoriously difficult to minimize, one often approximates L_0 norm with an L_1 norm. As shown in [29], the minimal L_1 norm solution often coincides with the sparsest solution for under-determined linear systems. We have successfully applied L_1 norm minimization to network anomaly inference [108].

As in [108], we solve the following variant of (2.29)

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_1 + \lambda \|\mathbf{x}\|_1. \quad (2.30)$$

An advantage of (2.30) is that it can be cast into an equivalent Linear Programming (LP) problem, for which solutions are available even with large-scale A , owing to modern interior-point LP methods. The LP formulation also allows one to incorporate additional linear constraints, such as the nonnegativity constraints $\mathbf{x} \geq 0$. Finally, if we have prior information that \mathbf{x} is close to an initial solution μ , we can replace $\|\mathbf{x}\|_1$ with $\|\mathbf{x} - \mu\|_1$ in (2.30), yielding

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_1 + \lambda \|\mathbf{x} - \mu\|_1. \quad (2.31)$$

2.3.2.3 Maximum Entropy Estimation

For inference under the nonnegativity constraints $\mathbf{x} \geq 0$, another commonly used solution is *maximum entropy estimation*, which uses the negative entropy function as $J(\mathbf{x})$ in (2.26).

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \sum_i x_i \log x_i, \quad \mathbf{x} \geq 0. \quad (2.32)$$

If we know \mathbf{x} is close to an initial solution $\mu = [\mu_1 \mu_2 \cdots \mu_n]^T$, we can instead minimize the relative entropy [23], resulting in

$$\min_{\mathbf{x}} \|\mathbf{y} - A\mathbf{x}\|_2^2 + \lambda^2 \sum_i x_i \log \frac{x_i}{\mu_i}, \quad \mathbf{x} \geq 0. \quad (2.33)$$

(2.33) can be efficiently solved by PDCO [85], which has been highly optimized for sparse matrices A . We have successfully applied (2.33) in the context of traffic matrix estimation [110].

2.3.3 Toolkit Development

We develop a toolkit on the PlanetLab [80] to measure and infer network path properties. Our toolkit is programmed in MATLAB, Perl, and C++, altogether with around 25,000 lines of code. The toolkit design is based on master-slave model. The master accepts measurement requests from users, designs measurement experiments, issues measurement commands to the slaves, and collects and analyzes the results. The slaves accept measurement commands from the master, conduct measurements, gather the results and return them back to the master. While our current implementation is based

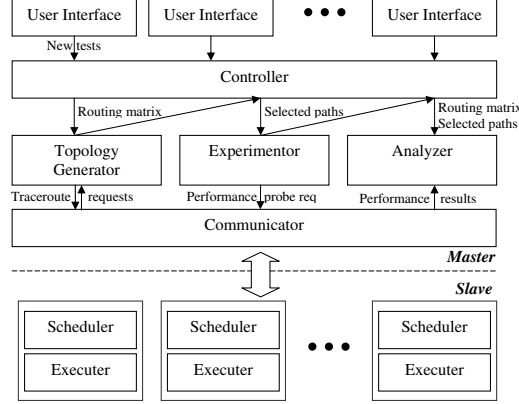


Figure 2.8: Our toolkit architecture.

on one master and multiple slaves, our architecture is extensible to multiple masters and multiple slaves.

As shown in Figure 2.8, the master consists of the following five modules: (i) controller, which accepts user measurement requests, schedules jobs, and manages the other master modules, (ii) topology manager, which generates and maintains the routing matrix, (iii) experimenter, which applies one of the experimental design algorithms in Section 2.3.1 to identify which paths to probe, and issues measurement requests to the corresponding slave nodes, (iv) analyzer, which applies one of the inference algorithms in Section 2.3.2 to infer the network performance based on the obtained measurement data, and (v) communication module, which takes care of communication between slaves and masters.

Slave side of the toolkit accepts measurement commands both from topology manager and experimenter. The request is queued at its scheduler, and executed according to the specified frequency and duration. When a set of measurement experiments has finished, the results are sent back to the communication module of the master. For safety and convenience, we use `scriptroute` [95] for conducting traceroute. The toolkit runs continuously to measure and infer performance on the paths specified by the users.

2.3.4 Evaluation

2.3.4.1 Accuracy Metric

We quantify the inference accuracy using normalized mean absolute error (MAE), which is defined as

$$\text{normalized } MAE = \frac{\sum_i |inferred_i - actual_i|}{\sum_i actual_i}, \quad (2.34)$$

where $inferred_i$ and $actual_i$ are the inferred and actual end-to-end performance for path i . A lower value of normalized MAE indicates better accuracy.

Another set of metrics we use to quantify the accuracy of loss inference, are false positive rate (FP) and false negative rate (FN). We consider a path is lossy if its loss rate is above a certain threshold. In our evaluation, a path with over 2% loss rate is considered as lossy. False positive rate refers to the fraction of paths that are inferred to be lossy but actually are not lossy, i.e., $Pr(inferred \text{ lossy} | actual \text{ not lossy})$. False negative rate refers to the fraction of paths that are inferred to be not lossy but are actually lossy,

	# nodes	# overlay nodes	# paths	# links	rank(A)
PlanetLab-RTT	2514	61	3657	5467	769
PlanetLab-Loss	1795	60	3270	4628	690

Table 2.4: Summary of PlanetLab traces used for evaluation.

i.e., $Pr(\text{inferred not lossy} | \text{actual lossy})$. As MAE, lower FN and FP indicate higher accuracy.

2.3.4.2 Dataset Description

We evaluate our approach using both real traces and synthetic data. Note that the real traces use round-trip performance measurements, whereas the synthetic data use one-way performance measurements. The problem formulation $\mathbf{y} = A\mathbf{x}$ works for both one-way and round-trip measurements.

PlanetLab traces: We collected RTT traces among PlanetLab using our toolkit on Oct. 1, 2005 for 10 minutes, with a probing frequency of one probe per second for every monitored path. We also collected loss traces on PlanetLab on Jan. 22, 2006 for 15 minutes, with a probing frequency of one probe per 300 milliseconds for every monitored path. Table 2.4 summarizes the traces, where *nodes* include both end hosts and intermediate routers on the end-to-end paths, and *overlay nodes* only include end hosts among which the end-to-end performance needs to be monitored or estimated.

We construct routing matrix, A , using traceroute measurements. We derive the actual RTT based on the mean over 60 probes in every 1-minute interval, and derive the actual loss rates based on the mean over 300 probes

in every 90-second interval. We use the following approach to derive the inferred RTT and loss rates: for the paths that are monitored we assume we know the true RTT and loss rates; for the un-monitored paths, we use the inference algorithms described in Section 2.3.2 to infer based on the observed performance of monitored paths. For each interval, we use normalized *MAE* to quantify the inference error.

Rocketfuel topologies: We also evaluate the performance of NetQuest on ISP topologies discovered by Rocketfuel [94, 93]. Specifically, we use two large router-level ISP backbone topologies as our underlay topologies: Sprint (in US), and Tiscali (in UK). Both topologies have been annotated with inferred link weights and link latencies as described in [62]. For each underlay topology, we randomly selected 60 routers to form an overlay. The route between two overlay nodes is computed as the shortest path on the underlay (using the inferred link weights). Table 2.5 summarizes the Rocketfuel topologies we use.

	# nodes	# overlay nodes	# paths	# links	rank(<i>A</i>)
Rocketfuel-Sprint	315	60	3540	622	518
Rocketfuel-Tiscali	161	60	3540	398	286

Table 2.5: Summary of Rocketfuel topologies used for evaluation.

To evaluate loss inference, we assign a loss rate to each link in the following way as in [78, 17, 112]. A fraction of links were classified as “good”, and the rest as “bad”. The loss rate for a good link is picked uniformly at random in the 0-1% range and that for a bad link is picked in the 5-10% range. Once each link has been assigned a loss rate, we derive the true loss rate for

each path. Then we use Bernoulli or Gilbert loss process at each path to derive the observed loss rate. In the Bernoulli model, each packet is dropped with a fixed probability determined by the loss rate of the path. In the Gilbert case, the path moves between a good state and a bad state, where no packets are dropped at the good state and all packets are dropped at the bad state. Following [72, 78], we use 35% as the probability of remaining in the bad state. The other state-transition probabilities are determined to match the average loss rate with the loss rate assigned to the link. In both cases, the end-to-end loss rate is computed based on the transmission of 10000 packets. Our evaluation shows that the inference accuracy is similar under Bernoulli and Gilbert loss models. So in the interest of brevity, we only show the results from Gilbert loss models.

Synthetic power-law topologies: To further test the scalability of our approach, we generate large synthetic power-law network topologies using BRITE topology generator [64]. Specifically, we generate two topologies using the router-level Barabasi-Albert model in BRITE (which is configured to have each new node connect to $m = 2$ existing nodes). Table 2.6 summarizes the BRITE topologies we use. The BRITE topology generator assigns each link with a delay based on its physical distance. We emphasize that we use large BRITE topologies primarily for illustrating the scalability of our approach as opposed to evaluating its performance in realistic scenarios. The PlanetLab traces and Rocketfuel topologies are used to achieve the latter goal.

	# nodes	# overlay nodes	# paths	# links	rank(A)
Brite-n1000-o200	1000	200	39800	2883	2051
Brite-n5000-o600	5000	600	359400	14698	9729

Table 2.6: Summary of BRITE topologies used for evaluation.

2.3.4.3 Experimental Parameters

There are several parameters in Bayesian experimental design and network inference. Below we present the values that we use for these parameters in our evaluation.

Prior information for Bayesian experimental design (R): Recall that the design criteria for both Bayesian A - and D -optimality are functions of $D(\eta) = (A_S A_S^T + R)^{-1}$. To estimate R , one needs to estimate both the variance of the measurement noise (σ^2) and the covariance matrix of the link performance ($\Sigma = \sigma^2 R^{-1}$) through network measurement. However, the estimation of such second-order statistics in large-scale network measurement can be both expensive and inaccurate (due to measurement artifacts and non-stationarities in Internet path properties).

In NetQuest, we avoid such difficulties by setting $R = \epsilon I$, where ϵ is a small constant and I is the identity matrix. Our results suggest that this simple choice of R yields designs that consistently out-perform the alternative designs we considered (see Section 2.3.4.4). Moreover, the resulting design is highly insensitive to the choice of ϵ . In our evaluation, we set $\epsilon = 0.001$, which yields good results. Finally, note that a similar approach has been taken in the literature of Bayesian super-saturated design [48].

Prior information for network inference (μ): In this work, unless noted otherwise, we assume no prior information about \mathbf{x} . That is, we set $\mu = \mathbb{0}$ (an all-0 vector) for L_2 and L_1 norm minimization (both with and without nonnegativity constraints), and $\mu = \mathbb{1}$ (an all-1 vector) for maximum entropy estimation. Despite not using any prior information, our results show that NetQuest can achieve high accuracy by probing only a small fraction of paths.

Regularization parameter (λ): Our experience [110, 108] suggests that the inference algorithms are not sensitive to the choice of λ . In our evaluation, we use $\lambda = 0.01$, which gives satisfactory results.

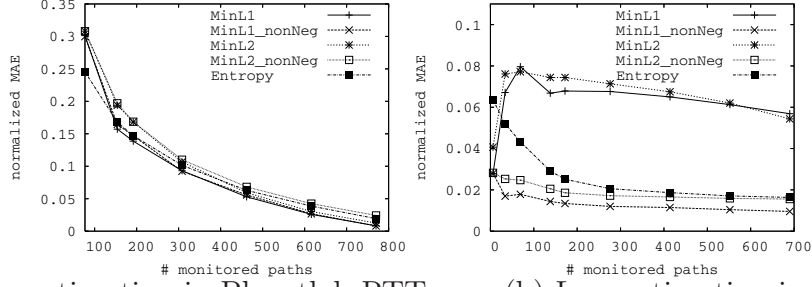
2.3.4.4 Basic Framework Evaluation Results

We first evaluate our basic measurement framework. Then we examine its capability of supporting flexible design requirements. Finally we study the effects of prior information.

In this section, we evaluate the two key components of our framework: (i) inference algorithms, and (ii) design of experiments.

Comparison of Inference Algorithms

First, we compare the accuracy of different inference algorithms. We use the A -optimal design criterion to determine which set of paths to monitor. Figure 2.9 (a) and (b) show the error of inferring end-to-end delay and loss rates as we vary the number of monitored paths. The x-value of the right most point on each curve corresponds to the rank of the routing matrix.



(a) Delay estimation in Planetlab-RTT. (b) Loss estimation in Planetlab-loss.

Figure 2.9: Comparison of inference algorithms for delay and loss estimation using A-optimal design.

As shown in Figure 2.9 (a), different inference algorithms perform similarly for delay inference. Moreover, as expected, the error decreases with an increasing number of monitored paths.

As shown in Figure 2.9 (b), the performance difference between various inference algorithms is larger for loss rate inference. The inference algorithms that enforce nonnegativity constraints out-perform those that do not enforce such constraints. In addition, the inference error under those algorithms without nonnegativity constraints does not decrease with an increasing number of monitored paths. Since loss rates take nonnegative values, intuitively enforcing nonnegativity constraints should give better inference. In comparison, the effect of nonnegativity constraints is much smaller for delay inference. This is because all paths have delay larger than 0, so even without enforcing nonnegativity constraints most links are assigned positive delay. Finally, MinL1 consistently out-performs the other inference schemes. As described in Section 2.3.2.2, MinL1 effectively maximizes the sparsity of link loss rate, which

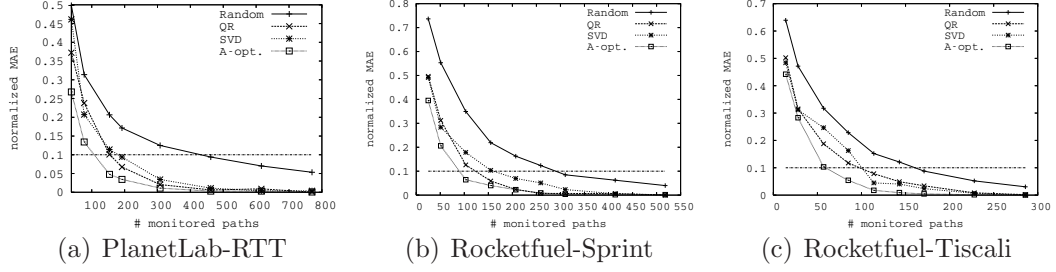


Figure 2.10: Comparison of experimental design schemes for estimating network-wide delay using MinL2 inference algorithm.

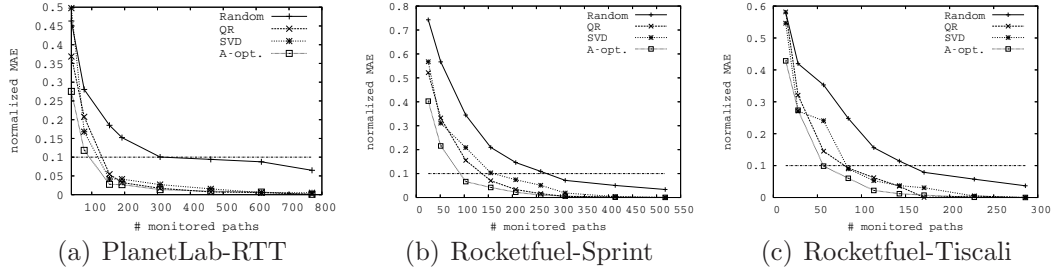


Figure 2.11: Comparison of experimental design schemes for estimating network-wide delay using MinL1_nonNeg inference algorithm.

matches well with the fact that few links on the Internet are lossy.

From the above results, in the rest of the chapter, unless noted otherwise, we use MinL1_nonNeg for both delay and loss inference.

Comparison of Measurement Designs

Next we evaluate different algorithms for designing measurement experiments. We consider inferring two types of quantities: network-wide performance and individual path performance.

First, we begin by addressing the inference of network-wide perfor-

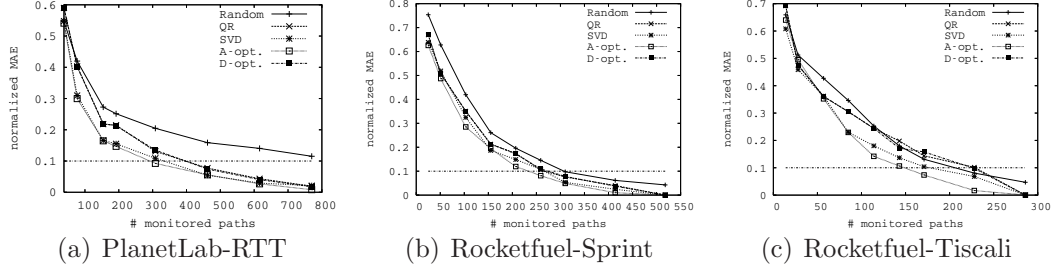


Figure 2.12: Comparison of experimental design schemes for per-path delay inference using MinL1_nonNeg inference algorithm.

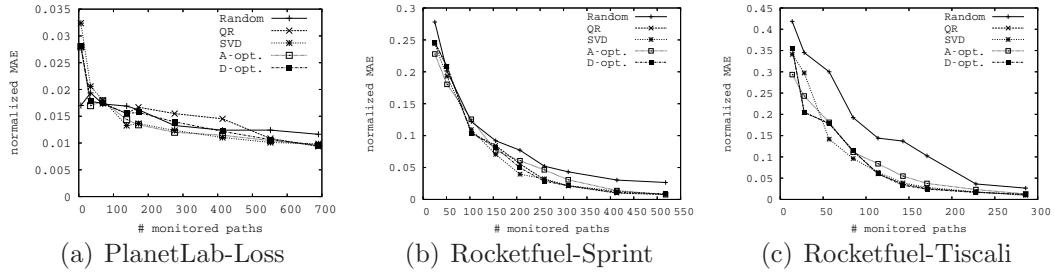


Figure 2.13: Comparison of experimental design schemes for per-path loss inference using MinL1_nonNeg inference algorithm (simulation uses Gilbert model).

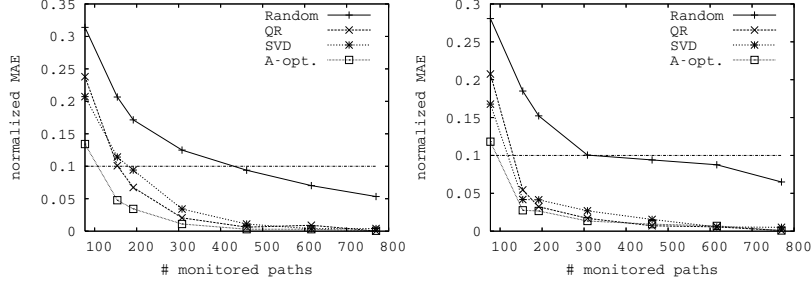
mance. A global view of the performance aggregated over an entire network is useful for a variety of reasons. It can be used for estimating a typical user experience (as in the Internet End-to-end Performance Monitoring Project (IEPM)), detecting anomalies, trouble-shooting, and optimizing performance. The pioneering work in this area is done by Chua et al. [18, 19], which is based on SVD.

We compare the Bayesian experimental designs with the other alternatives for inferring network wide average delay. Here the quantity of interest is

$f(\mathbf{x}) = \frac{1}{m} \mathbb{1} A \mathbf{x}$, where $\mathbb{1}$ is an all-1 row vector of length m . We use the PlanetLab traces to evaluate the performance under a realistic scenario, and use simulated topologies to further test the scalability of our measurement design. Figure 2.14 (a) compares different experimental design schemes when MinL2 is used for inference, and Figure 2.14 (a) shows the results when MinL1_nonNeg is used. As noted in Section 2.3.1.1, A -optimal and D -optimal designs are identical for inferring network-wide average, so we only show the results of A -optimal.

As shown Figure 2.14 (a), with the A -optimal design, the inference error decays very fast – the error is within 15% by monitoring only within 2% paths (e.g., 77 out of 3657 paths in PlanetLab-RTT). In comparison, the inference error is 50% or higher (than A -optimal) when the same number of paths are monitored under the other schemes. In addition, we observe that to achieve within 10% inference error, the other schemes require monitoring 60% more paths than A -optimal. Finally, as the number of monitored paths increases, all the schemes converge to close to 0 inference error, since in this case there are sufficient information to reconstruct the global network view. Random selection converges slower because it does not ensure the selected paths are linearly independent. Similar results are observed in Figure 2.14 (b) when the inference algorithm changes to MinL1_nonNeg.

Second, we now address the inference of individual path performance. Figure 2.12 compares different measurement design schemes for inferring individual path delay. As we can see, the rank-based approach requires monitor-



(a) MinL2 inference algorithm

(b) MinL1 inference algorithm

Figure 2.14: Comparison of experimental designs for estimating network-wide delay.

ing 286–786 paths, which is expensive. In comparison, the other approaches can provide a smooth tradeoff between inference accuracy and measurement overhead. Among them, *A*-optimal consistently performs the best, though the benefit becomes smaller compared with network-wide performance inference. Note that *D*-optimal performs significantly worse than *A*-optimal, and sometimes even worse than the other alternatives. This suggests that the Kullback-Leibler distance tends to under-penalize estimation errors.

Figure 2.13 shows the absolute inference error in loss rate estimation. *A*-optimal performs very close to the other schemes. The performance difference is much smaller than that of delay, because most links have close to zero loss rate, and assigning links with zero loss rate (even without extensive network monitoring) can achieve low average inference error. Note that when the number of monitored paths is equal to the rank of the routing matrix, the error is non-zero due to the sampling errors when assigning true loss rates in Rock-ETFuel topologies. Finally, we observe that *D*-optimal often performs worse

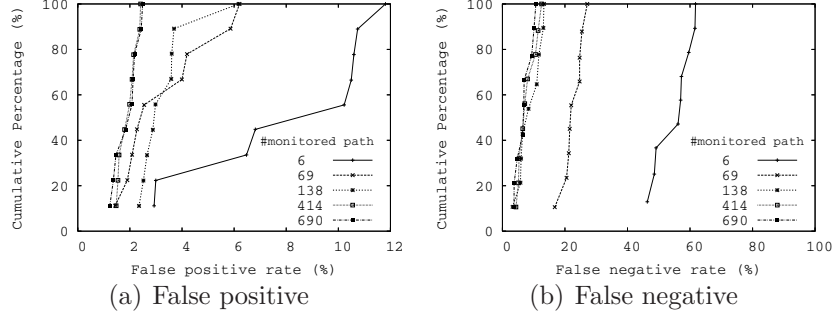


Figure 2.15: False positive and false negative rates of per-path loss inference using MinL1_nonNeg inference algorithm in PlanetLab-Loss.

than or similar to A -optimal. We observe similar results for Bernoulli loss model and other topologies. So hereafter we will focus on Bayesian A -optimal designs.

We further quantify the accuracy of loss estimation using false positive and false negative rates. As shown in Figure 2.15(a) and (b), the average false positive and false negative rates are within 5% and 22%, respectively, even when the number of monitored paths is 10% of the rank of the routing matrix. Moreover, we observe that both false positive and false negative rates decrease as the number of monitored paths increases.

To illustrate the scalability of our approach, Figure 2.16 compares the performance of different measurement design schemes on the large synthetic BRITE topologies. Our results show that A -optimal is more scalable than SVD and QR, both of which cannot handle Brite-n5000-o600, and fail to run on Brite-n1000-o200 when the number of monitored paths exceeds 1200.

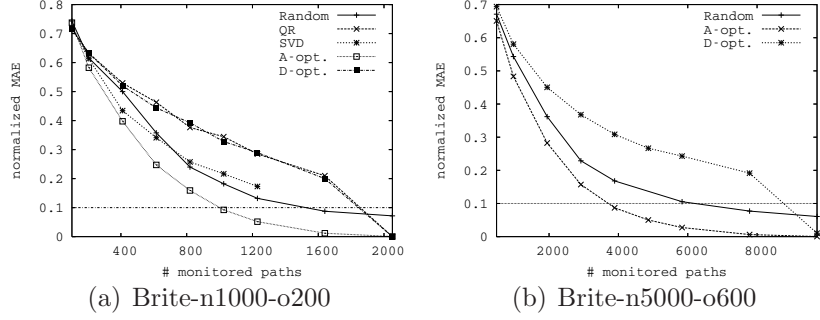


Figure 2.16: Comparison of experimental design schemes for per-path delay inference using MinL1_nonNeg inference algorithm on large BRITE topologies.

Therefore the SVD and QR curves in Figure 2.16(a) stop at 1200 monitored paths, and Figure 2.16(b) only shows the results for Bayesian experimental designs and random path selection for Brite-n5000-o600. In terms of accuracy, *A*-optimal consistently outperforms the other design schemes.

Summary

In this section we evaluate our measurement framework. Our key findings are:

- Design of measurement experiments is crucial for large-scale network monitoring. *A*-optimal is effective in constructing measurement experiments for inferring network-wide average delay. It can achieve 15% inference error by monitoring only 2% paths. Moreover it is also competitive for estimating individual path performance. Compared with the existing approach, SVD, *A*-optimal achieves comparable accuracy for estimating individual path properties and achieves higher accuracy in

estimating network-wide average delay. In addition, A-optimal is much more scalable than SVD, and can be applied to networks with thousands of routers and end hosts. Furthermore, as we will show in the next section, A-optimal design provides high flexibility, which is another major advantage over the existing SVD approach.

- Our results show that the different inference algorithms under study perform similarly for delay inference, whereas the algorithms that enforce non-negativity constraints perform better for loss inference.

2.3.4.5 Flexibility of Measurement Design

In this section, we evaluate the flexibility of our measurement framework by estimating delay on individual network paths in the PlanetLab-RTT topology.

Differentiated Design First we examine the effectiveness of experimental designs for providing differentiated treatment to a subset of paths. We apply the technique described in Section 2.3.1.2 to achieve this goal. In our evaluation, we randomly assign a subset of preferred paths with a weight varying from 1 to 16, while fixing the weight on the remaining paths to 1. Figure 2.17 shows the inference error on both the preferred and the remaining paths when we monitor 200 paths in PlanetLab-RTT topology and vary the number of preferred paths from 20 to 160. We make the following observations. First, as we would expect, the inference error on the preferred paths decreases with an increasing weight. When the weight is 4 and higher, the inference error is

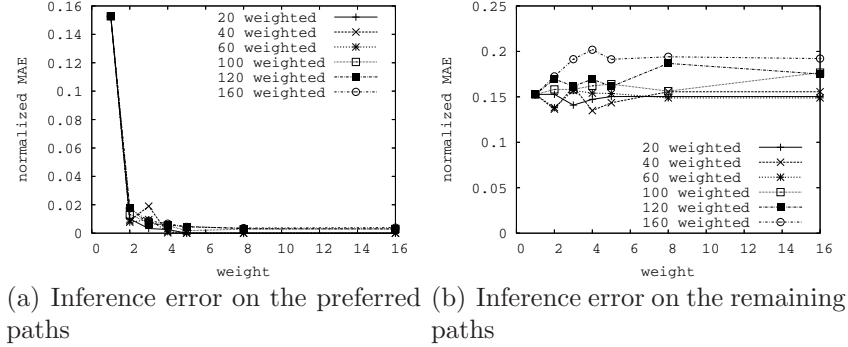


Figure 2.17: Use differentiated design based on A -optimal design to provide a higher resolution in estimating a subset of preferred paths in PlanetLab-RTT.

close to 0. This is because when the weight is high enough, the performance of many preferred paths is either directly monitored or exactly reconstructed from the monitored paths. Second, we observe that the inference error on the remaining paths increases slightly, because as we pay more attention to the preferred paths, the remaining paths are monitored less extensively. For a similar reason, the inference error of the remaining paths tends to increase slightly with an increasing number of preferred paths.

2.3.4.6 Augmented Design

Next we consider augmented design for supporting continuous monitoring. Our evaluation is based on the following scenario. Suppose we identify a set of paths to monitor, and some of them fail to provide us measurement data (*e.g.*, due to software or hardware failures at monitor sites or at their incoming/outgoing links). In this case, we need to identify the additional

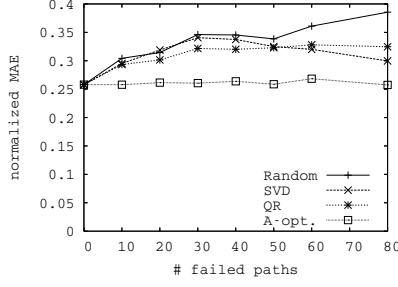


Figure 2.18: Comparison of augmented design schemes in PlanetLab-RTT.

measurements to conduct given that we have already obtained the measurement results from the unfailed paths.

In our evaluation, we first use A -optimal to identify 100 paths to monitor in PlanetLab-RTT. Then we vary the number of failed paths from 10 to 80, and apply different augmented design algorithms to determine the additional paths to monitor. Figure 2.18 shows the average inference error under different schemes. As we can see, A -optimal yields the lowest error. Moreover, its inference error is similar for a varying number of failed paths. In comparison, the inference error of the other schemes tends to increase with an increasing number of failed paths. This suggests that the A -optimal design is the most effective in augmenting existing designs.

Multi-user Design Now we study the multi-user scenarios, where each user is interested in a certain part of network. We compare the following design schemes:

- *Separate design and separate inference (sep./sep.)*: Each user individually determines the set of paths to monitor, and makes inference based

solely on his/her own observations.

- *Separate design and joint inference (sep./joint)*: This is an enhancement of the previous version. Users still individually decide which paths to monitor, but they make inference based on the observations made from all users.
- *Augmented design and joint inference (aug./joint)*: In the augmented design, we first design measurement experiments for user 1, and then apply the augmented design (in Section 2.3.4.6) to construct measurement experiments for user 2. We continue the process for all the other users.
- *Union design and joint inference (union/joint)*: In the union design, we take a union of all the paths that are interesting to at least one user, and then apply the (basic) measurement design.
- *Joint design and joint inference (joint/joint)*: Unlike in the union design, where all interesting paths are treated equally, in joint design we set a path's weight to be the square root of the number of users who are interested in the path (see Section 2.3.1.2).

All the design algorithms can work in separate, augmented, and union modes. In addition, A -optimal can also support joint design.

In our evaluation, we have 16 users, each interested in 50 paths. There are a common set of paths that are interesting to all users. Figure 2.19(a) com-

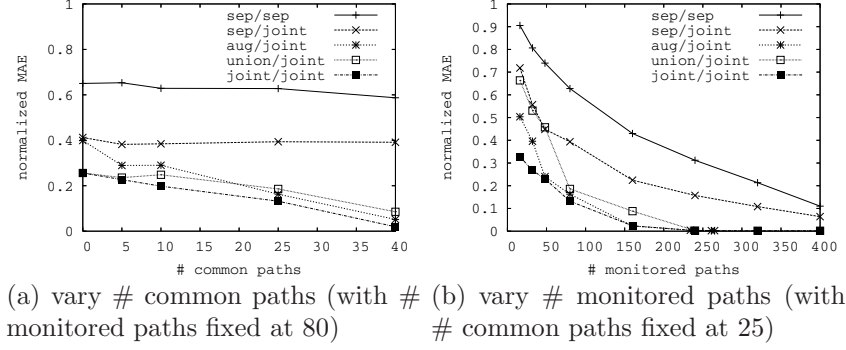


Figure 2.19: Comparison of different design modes in handling multi-user scenarios using PlanetLab-RTT, where all modes use the A -optimal design.

compares the A -optimal design in its various design modes for inferring individual path delay as the number of common paths is varied from 0 to 40. The remaining paths interesting to a user are randomly selected from all the non-common paths. In the order of accuracy ranking, $\text{joint./joint} > \text{union./joint} \approx \text{aug./joint} > \text{sep./joint} > \text{sep./sep.}$. In particular, sep./sep. incurs significantly higher error than the others, because in this case different users do not share their observation. Enabling information sharing in sep./joint reduces the normalized MAE by 0.2 or higher. A further reduction is achieved by incorporating users' interest into measurement design. Interestingly, augmented design performs similarly to union design, even though the former is an online version of the latter (*i.e.*, the i -th user determines its measurement experiments without considering the j -th user's interest for $j > i$). The performance of joint./joint is even better, and its benefit over separate design grows as the number of common paths increases. This is attributed to the fact that it not only incorporates

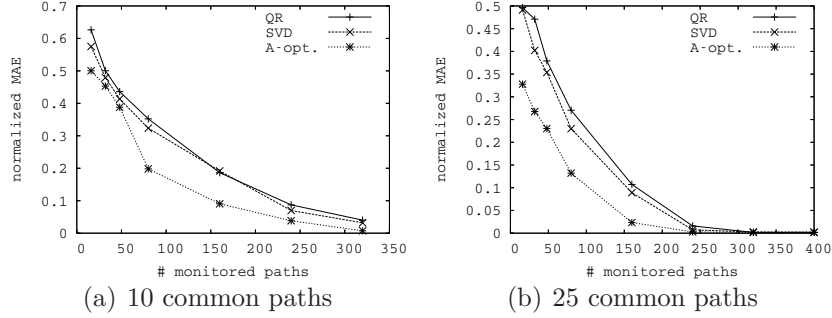


Figure 2.20: Comparison of different design schemes using their best modes on PlanetLab-RTT.

all users' interest in designing measurement, but also it biases measurement towards paths that interest more users. Figure 2.19(b) compares the inference error as we vary the number of monitored paths. As before, joint/joint yields the best performance among all the schemes. The performance gap is largest with a small number of monitored paths. This suggests that experimental design is especially important under tight measurement resource constraints.

Next we compare the performance across different design schemes. Figure 2.20 (a) and (b) show the inference error of various design schemes under their best design modes. They all use joint inference. As they show, *A-optimal* yields the lowest error, up to 80% lower than the alternatives.

Summary

To summarize, we demonstrate the flexibility of our measurement framework using the real trace. Our results show that it can effectively support differentiated design, augmented design, and joint design. Such capabilities

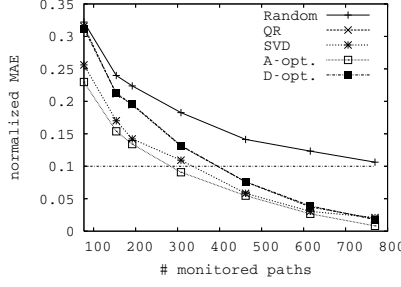


Figure 2.21: Effects of the enhanced prior on PlanetLab-RTT.

are useful for a variety of network monitoring applications.

2.3.5 Effects of Prior Information

So far we assume no prior information about \mathbf{x} . In our future work, we plan to develop light-weight techniques to obtain better priors and thus further improving the inference accuracy. As a first step, below we develop a simple method for obtaining an enhanced prior. Specifically, we consider a special form of prior whose elements are all equal: $\mu = z \cdot \mathbf{1}$, where z is an unknown, and $\mathbf{1}$ is an all-1 column vector of length n . We can estimate z by solving an over-determined least-squares problem: $\mathbf{y} = A\mu = (A\mathbf{1})z$, yielding $z = \frac{(A\mathbf{1})^T \mathbf{y}}{\|A\mathbf{1}\|_2^2}$. Intuitively, the resulting prior $\mu = \frac{(A\mathbf{1})^T \mathbf{y}}{\|A\mathbf{1}\|_2^2} \cdot \mathbf{1}$ estimates the average link performance. An advantage of this method is that it is extremely simple and requires no extra measurement.

Figure 2.21 shows the accuracy of delay inference for different measurement design schemes using the above enhanced prior. Compared with Figure 2.12(a), we make the following observations. First, the enhanced prior improves the inference accuracy for all measurement design schemes. It re-

duces the normalized MAE by up to 0.07 (or about 25%). The improvement is largest when the number of monitored paths is small. This is because the enhanced prior information is most helpful to compensate for incomplete monitoring information. In comparison, with extensive monitoring we can accurately estimate performance even without prior. Second, the relative ranking of different design schemes remains the same as before. A-optimal design continues to yield the highest accuracy.

On the other hand, the enhanced prior only yields very little accuracy improvement for loss inference (results omitted in the interest of brevity). This is not surprising, because most links have very low loss rate, making $\mu = \mathbb{0}$ a good prior.

2.4 Summary of Network Property Inference

In this chapter, we develop two path performance measurement and analysis schemes: exact reconstruction of path performance and approximate reconstruction. In the former, we propose a scalable algorithm to overlay network monitoring which provides an upper-bound guarantee of amount of active probing for full reconstruction of path properties.

In the latter, we build a framework that leverages powerful tools developed in the field of Bayesian experimental design. The framework is highly scalable and can design measurement experiments that span thousands of routers and end hosts. It also is flexible to accommodate a variety of design objectives from different administrative entities.

Chapter 3

Scalable and Accurate Social Network Analysis

A *social network* [103] is a social structure modeled as a graph, in which nodes represent people or other entities embedded in a social context, and edges represent specific types of interdependency among entities, such as values, visions, ideas, financial exchange, friendship, kinship, dislike, conflict, or trade. Understanding the nature and evolution of social networks has important applications in a number of fields such as sociology, anthropology, biology, economics, information science, and computer science.

Traditionally, studies of social networks often focus on relatively small social networks (*e.g.*, [56, 57] examine co-authorship networks with about 5000 nodes). Recently, however, social networks have gained tremendous popularity in the cyber space. Online social networks such as MySpace [70], Facebook [32] and YouTube [105] have each attracted tens of millions of visitors every month [77] and are now among the most popular sites on the Web [3]. The wide variety of online social networks and the vast amount of rich information available in these networks represent an unprecedented research opportunity for understanding the nature and evolution of social networks at massive scale.

A central concept in the computational analysis of social networks is *proximity measure*, which quantifies the closeness or similarity between nodes in a social network. Proximity measure lies at the heart of many important applications, ranging from viral marketing to evolutionary study of human networks. Given its importance, a variety of proximity measures have been proposed. The simplest proximity measures are based on either the shortest graph distance or the maximum information flow between two nodes. One can also define proximity measures based on node neighborhoods (*e.g.*, the number of common neighbors). Finally, several more sophisticated proximity measures involve infinite sums over the ensemble of all paths between two nodes (*e.g.*, Katz measure [49], rooted PageRank [56, 57], and escape probability [99]). Compared with more direct proximity measures such as shortest graph distances and numbers of shared neighbors, path-ensemble based proximity measures incorporate more information about the underlying social structure and have been shown to be more effective in social networks with thousands of nodes [56, 57, 99].

Unfortunately, the explosive growth of online social networks imposes significant challenges on proximity estimation. First, online social networks are typically *massive in scale*. For example, MySpace has over 400 million user accounts [71], and Facebook has reportedly over 120 million *active* users world wide [33]. As a result, many proximity measures that are highly effective in relatively small social networks (*e.g.*, the classic Katz measure [49]) become computationally prohibitive in large online social networks with millions of

nodes [84]. Second, online social networks are often *highly dynamic*, with hundreds of thousands of new nodes and millions of edges added daily. In such fast-evolving social networks, it is challenging to compute up-to-date proximity measures in a timely fashion.

In this chapter, we address the above challenges by presenting two *scalable* and *flexible* proximity approximation techniques to approximate a large family of path-ensemble based proximity measures.

Scalable Proximity Inversion.

We first present *scalable proximity inversion* [87], our initial progress on approximating a restricted sub-family of path ensemble based proximity measures on social networks with millions of nodes. We present our techniques applied to a large family of commonly used proximity measures, which includes the aforementioned Katz measure [49], as well as rooted PageRank [56, 57] and escape probability [99].

Clustered Spectral Graph Embedding.

While Scalable Proximity Inversion makes significant progress in approximating proximity measures that form low-rank matrices, there is still room for improvement to handle proximity measures *without* low-rankness. As a follow-up of Scalable Proximity Inversion, we propose *Clustered Spectral Graph Embedding(CSGE)* [91], which dramatically improve the accuracy and scalability of proximity estimation over Proximity Embedding and Proximity Sketch. Clustered spectral graph embedding embeds the original highly sparse

but massive social graph into a dense but much smaller graph. The embedded graph captures the essential clustering and spectral structure of the original graph, and allows a wide range of analysis tasks to be performed in an efficient and accurate fashion. The technique is also flexible, and can naturally support incremental update and parallel/distributed computation, which is essential for social networks that are highly dynamic and/or decentralized.

We use three large real-world social network datasets (LiveJournal [61], Flickr [36] and MySpace [70] with up to 2 million nodes and 90 million links) to experimentally demonstrate the effectiveness of our approach. In particular, we evaluate its accuracy, scalability and flexibility in the context of three important social network analysis tasks: (i) *proximity estimation* (*i.e.*, approximating well-known proximity measures proposed in the literature), (ii) *missing link inference* (*i.e.*, inferring the locations of unobserved links based on observed links), and (iii) *link prediction* (*i.e.*, predicting which node pairs will become connected based on past snapshots of the social network).

In the context of proximity estimation, our new techniques results in nearly an order of magnitude speedup over the state-of-the-art proximity estimation techniques [88]. More importantly, with the same memory requirement, our technique is able to create approximations when the rank is an order of magnitude higher than previous methods. As a result, our technique results in dramatic improvement on the approximation accuracy of proximity measures, such as rooted PageRank and escape probability, which are not low-rank and thus cannot be approximated accurately by previous methods. In the con-

text of missing link inference, our technique results in several-fold reduction in the false positive rate subject to the same false negative rate. In the context of link prediction, our technique yields a novel *supervised proximity measure* that significantly improves the link prediction accuracy. These results clearly demonstrate the effectiveness and the practical values of our approach.

3.1 Problem Formulation

A social network is denoted as a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{1, 2, \dots, |\mathcal{V}|\}$ is a set of vertices, and $\mathcal{E} = \{e_{ij} \mid i, j \in \mathcal{V}\}$ is a set of edges. In particular, if there is an edge between vertex i and vertex j , then e_{ij} denotes the weight of this edge. The *adjacency matrix* A of the graph G is an $m \times m$ matrix with $m = |\mathcal{V}|$:

$$a_{ij} = A[i, j] = \begin{cases} e_{ij}, & \text{if there is an edge between } i \text{ and } j, \\ 0, & \text{otherwise.} \end{cases}$$

Below we formally define some of the most commonly used proximity measures and show that all of them can be efficiently estimated by solving a subproblem, the *proximity inversion problem*.

3.1.1 Proximity measures

Proximity measures are important for many social network applications. Most proposed proximity measures can be divided into two broad categories: (1) direct measures that are based on shortest graph distances or the maximum information flow between two nodes or node neighborhood, *e.g.*,

common neighbors; and (2) more sophisticated measures that include infinite sums over ensembles of all paths between two nodes, *e.g.*, *Katz measure* [49], *rooted PageRank* [57], and *escape probability* [99]. It has been shown that the path-ensemble based proximity measures capture much more information about the underlying network compared to the direct measures and are generally more effective for various tasks [57, 99]. Now, we define the proximity measures mentioned above.

Common neighbors. Let N_i be the neighbor set of vertices i . Then, the common neighbors proximity measure is

$$P_{\text{cn}}[i, j] = |N_i \cap N_j|.$$

If the number of common neighbors is high between vertices i and j , then it is more likely they will get connected. For an *unweighted* graph (*i.e.*, $e_{ij} = 1$ for all edges), the common neighbor for all vertex pairs is simply given by $P_{\text{cn}} = A^2$.

Katz measure. Let $p_{ij}^{(k)}$ denote the number of paths of length k between vertices i and j . Then, the Katz measure is

$$P_{\text{kz}}[i, j] = \sum_{k=1}^{\infty} \beta^k p_{ij}^{(k)},$$

where $\beta < 1$ is a damping parameter. A high value of the Katz measure between two vertices signifies a stronger relationship. Using the adjacency matrix A , we may write the Katz measure for all vertex pairs simultaneously as

$$P_{\text{kz}} = \sum_{k=1}^{\infty} \beta^k A^k = (I - \beta A)^{-1} - I,$$

where I is an identity matrix.

Rooted PageRank. The rooted PageRank measure is the stationary probability that vertex j returns to vertex i in a random walk with a probability α that j jumps to i in each step, and with a probability $1 - \alpha$ that the process continues to a random neighbor. Let D be the diagonal *degree matrix* given by $D[i, i] = \sum_j A[i, j]$. Let $T = D^{-1/2}AD^{-1/2}$ be the normalized adjacency matrix. The stationary probability of the rooted PageRank for all vertex pairs is given by

$$\begin{aligned} P_{\text{rpr}} &= (1 - \alpha)(I - \alpha D^{-1}A)^{-1} = (1 - \alpha)D^{-1/2}(I - \alpha T)^{-1}D^{1/2} \\ &= (1 - \alpha)D^{-1/2}\left(\sum_{k=0}^{\infty} \alpha^k T^k\right)D^{1/2}. \end{aligned}$$

Escape probability. The escape probability is the probability in a random walk that vertex i will visit the vertex j before it returns to i . The measure is given by

$$P_{\text{ep}}[i, j] = f(P_{\text{rpr}}, i, j),$$

where the function f is defined as

$$f(P, i, j) = \frac{(1 - \alpha)P[i, j]}{P[i, i]P[j, j] - P[i, j]P[j, i]}. \quad (3.1)$$

The proximity inversion problem. From the above discussions, it is evident that the key to estimating all three path-ensemble based proximity measures is to efficiently compute elements of the following matrix inverse:

$$P \triangleq (I - \gamma M)^{-1} = \sum_{\ell=0}^{\infty} \gamma^\ell M^\ell \quad (3.2)$$

where M is a sparse nonnegative matrix with millions of rows and columns, I is an identity matrix of the same size, and $\gamma \geq 0$ is a damping factor (*e.g.*, $\gamma == \beta$ in Katz, and $\gamma == \alpha$ in rooted PageRank and escape probability). We term this common subproblem the *proximity inversion problem*.

3.2 Scalable Proximity Inversion

As a first step towards scalable and flexible proximity approximation, we propose *Scalable Proximity Inversion* [87]. The key challenge in solving the proximity inversion problem (*i.e.*, computing elements of matrix $P = (I - \gamma M)^{-1}$) is that while M is a sparse matrix, P is a dense matrix with millions of rows and columns. It is thus computationally prohibitive to compute and/or store the entire P matrix. To address the challenge, we first develop two novel dimensionality reduction techniques to approximate elements of $P = (I - \gamma M)^{-1}$ based on a static snapshot of M : *proximity sketch* and *proximity embedding*.

3.2.1 Preparation

We first present an algorithm to approximate the sum of a subset of rows or columns of $P = (I - \gamma M)^{-1}$ efficiently and accurately. We use this algorithm as a basic building block in both proximity sketch and proximity embedding.

Algorithm. Suppose we want to compute the sum of a subset of columns: $\sum_{i \in S} P[*, i]$, where S is a set of column indices. We first construct an indicator

column vector v such that $v[i] = 1$ for $\forall i \in S$ and $v[j] = 0$ for $\forall j \notin S$. The sum of columns $\sum_{i \in S} P[*, i]$ is simply Pv and can be approximated as:

$$Pv = (I - \gamma M)^{-1} v = \sum_{\ell=0}^{\infty} \gamma^{\ell} M^{\ell} v \approx \sum_{\ell=0}^{\ell_{\max}} \gamma^{\ell} M^{\ell} v \quad (3.3)$$

where ℓ_{\max} bounds the maximum length of the paths over which the summation is performed.

Similarly, to compute the sum of a subset of rows $\sum_{i \in S} P[i, *]$, we first construct an indicator row vector u such that $u[i] = 1$ for $\forall i \in S$ and $u[j] = 0$ for $\forall j \notin S$. We then approximate the sum of rows $\sum_{i \in S} P[i, *] = uP$ as:

$$uP = u(I - \gamma M)^{-1} = \sum_{\ell=0}^{\infty} \gamma^{\ell} u M^{\ell} \approx \sum_{\ell=0}^{\ell_{\max}} \gamma^{\ell} u M^{\ell} \quad (3.4)$$

In one extreme where S contains all the column indices, we can compute the sum of all columns in P . This is useful for computing the PageRank (which is the average of all columns in the RPR matrix). In the other extreme where S contains only one element, we can efficiently approximate a single row or column of P .

Complexity. Suppose M is an m -by- m matrix with n non-zeros. Computing the product of sparse matrix M and a dense vector v takes $O(n)$ time by exploiting the sparseness of M . So it takes $O(n \cdot \ell_{\max})$ time to compute $\{M^{\ell}v \mid \ell = 1, \dots, \ell_{\max}\}$ and approximate Pv . Note that the time complexity is independent of the size of the subset S . The complexity for computing uP is identical.

$P[x,y]$ is hashed into entry $S_k[x, g_k(y)]$ in each hash table S_k ($k=1, \dots, H$)
 So $S_k[x, g_k(y)]$ gives an upper bound on $P[x,y]$
 Estimate $P[x,y]$ by taking the min upper bound in all H hash tables

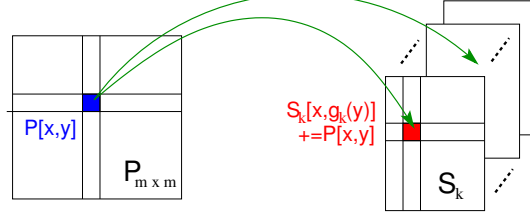


Figure 3.1: Proximity sketch

Note however that the above approximation algorithm is *not* efficient for estimating individual elements of P . In particular, even if we only want a single element $P[x,y]$, we have to compute either a complete row $P[x,*]$ or a complete column $P[*,y]$ in order to obtain an estimate of $P[x,y]$. So we only apply the above technique for preprocessing. We will develop several techniques in the rest of this section to estimate individual elements of P efficiently.

Benefits of truncation. We achieve two key benefits by truncating the infinite expansion $\sum_{\ell=0}^{\infty} \gamma^{\ell} M^{\ell}$ to form a finite expansion $\sum_{\ell=0}^{\ell_{\max}} \gamma^{\ell} M^{\ell}$. First, we completely eliminate the influence of paths with length above ℓ_{\max} on the resulting sums. This is desirable because as pointed out in [56, 57], proximity measures that are unable to limit the influence of overly lengthy paths tend to perform poorly for link prediction. Second, we ensure that $\sum_{\ell=0}^{\ell_{\max}} \gamma^{\ell} M^{\ell}$ is always finite, whereas elements of $\sum_{\ell=0}^{\infty} \gamma^{\ell} M^{\ell}$ may reach infinity when the damping factor γ is not small enough.

3.2.2 Proximity Sketch

Our first dimensionality reduction technique, *proximity sketch*, exploits the mice-elephant phenomenon that frequently arises in matrix P in practice, *i.e.*, most elements in P are tiny (*i.e.*, mice) but few elements are huge (*i.e.*, elephants).

Algorithm. Figure 3.1 shows the data structure for our proximity sketch, which consists of H hash tables: S_1, \dots, S_H . Each S_k is a 2-dimensional array with m rows and $c \ll m$ columns. A column hash function $g_k : \{1, \dots, m\} \rightarrow \{1, \dots, c\}$ is used to hash each element in $P_{m \times m}$ ($P[x, y]$) into an element in S_k ($S_k[x, g_k(y)]$). We ensure that different hash functions $g_k(\cdot)$ ($k = 1, \dots, H$) are two-wise independent. In each S_k , each element $P[x, y]$ is added to entry $S_k[x, g_k(y)]$. Thus,

$$S_k[a, b] = \sum_{y: g_k(y)=b} P[a, y] \quad (3.5)$$

Note that each column of S_k : $S_k[:, b] = \sum_{y: g_k(y)=b} P[:, y]$ can be computed efficiently as described in Section 3.2.1.

Since P is a nonnegative matrix, for any $x, y \in V$ and any $k \in [1, H]$, $S_k[x, g_k(y)]$ is an upper bound for $P[x, y]$ according to Eq. 3.5. We can therefore estimate $P[x, y]$ by taking the minimum upper bound in all H hash tables in $O(H)$ time. That is:

$$\hat{P}[x, y] = \min_k S_k[x, g_k(y)] \quad (3.6)$$

Probabilistic accuracy guarantee. Our proximity sketch effectively summarizes each row of P : $P[x, *]$ using a *count-min sketch* [21]: $\{S_k[x, *] \mid k = 1, \dots, H\}$. As a result, we provide the same probabilistic accuracy guarantee as the count-min sketch, which is summarized in the following theorem (see [21] for detailed proof).

Theorem 1. *With $H = \lceil \ln \frac{1}{\delta} \rceil$ hash tables, each with $c = \lceil \frac{\epsilon}{\delta} \rceil$ columns, the estimate $\hat{P}[x, y]$ guarantees: (i) $P[x, y] \leq \hat{P}[x, y]$; and (ii) with probability at least $1 - \delta$, $\hat{P}[x, y] \leq P[x, y] + \epsilon \cdot \sum_z P[x, z]$.*

Therefore, as long as $P[x, y]$ is much larger than $\epsilon \cdot \sum_z P[x, z]$, the relative error of $\hat{P}[x, y]$ is small with high probability.

Extension. If desired, we can further reduce the space requirement of proximity sketch by aggregating the rows of S_k (at the cost of lower accuracy). Specifically, we associate each S_k with a row hash function $f_k(\cdot)$. We then compute

$$R_k[a, b] = \sum_{x: f_k(x)=a} S_k[x, b] \quad (3.7)$$

and store $\{R_k\}$ (instead of $\{S_k\}$) as the final proximity sketch. Clearly, we have $R_k[a, b] = \sum_{x: f_k(x)=a} \sum_{y: g_k(y)=b} P[x, y]$. For any $x, y \in V$, we can then estimate $P[x, y]$ as

$$\hat{P}[x, y] = \min_k R_k[f_k(x), g_k(y)] \quad (3.8)$$

3.2.3 Proximity Embedding

Our second dimensionality reduction technique, *proximity embedding*, applies matrix factorization to approximate P as the product of two rank- r factor matrices U and V :

$$P_{m \times m} \approx U_{m \times r} \cdot V_{r \times m} \quad (3.9)$$

In this way, with $O(2mr)$ total state for factor matrices U and V , we can approximate any $P[x, y]$ in $O(r)$ time as:

$$\hat{P}[x, y] = \sum_{k=1}^r U[x, k] \cdot V[k, y] \quad (3.10)$$

Our technique is motivated by recent research on embedding network distance (*e.g.*, end-to-end round-trip time) into low-dimensional space (*e.g.*, [75, 58, 97, 63]). Note however that proximity is the opposite of distance — the lower the distance the higher the proximity. As a result, techniques effective for distance embedding do not necessarily work well for proximity embedding.

Algorithm. As shown in Figure 3.2(a), our goal is to derive the two rank- r factor matrices U and V based on only a subset of rows $P[L, *]$ and columns $P[*, L]$, where L is a set of indices (which we term the landmark set). We achieve this goal by taking the following five steps:

1. Randomly select a subset of ℓ nodes as the landmark set L . The probability for a node i to be included in L is proportional to the PageRank of node i

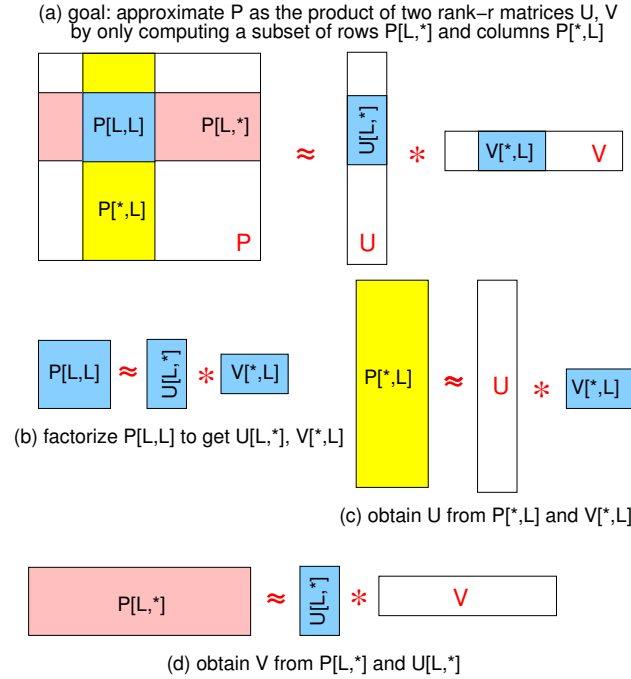


Figure 3.2: Proximity embedding

in the underlying graph¹. Note that the PageRank for all the nodes can be precomputed efficiently using the finite expansion method in Section 3.2.1.

2. Compute sub-matrices $P[L, *]$ and $P[*, L]$ efficiently by computing each row $P[i, *]$ and each column $P[*, i]$ ($i \in L$) separately as described in Section 3.2.1.
3. As shown in Figure 3.2(b), apply singular value decomposition (SVD) to obtain the best rank- r approximation of $P[L, L]$:

$$P[L, L] \approx U[L, *] \cdot V[*, L] \quad (3.11)$$

¹We also consider uniform landmark selection, but it yields worse accuracy than PageRank based landmark selection (see Section 4.3).

4. Our goal is to find U and V such that $U \cdot V$ is a good approximation of P . As a result, $U \cdot V[* , L]$ should be a good approximation of $P[* , L]$. We can therefore find U such that $U \cdot V[* , L]$ best approximates sub-matrix $P[* , L]$ in least-squares sense (shown in Figure 3.2(c)). Given the use of SVD in step 3, the best U is simply

$$U = P[* , L] \cdot V[* , L]^T \cdot (V[* , L] \cdot V[* , L]^T)^{-1} \quad (3.12)$$

5. Similarly, find V such that $U[L , *] \cdot V$ best approximates sub-matrix $P[L , *]$ in least-squares sense (shown in Figure 3.2(d)), which is simply

$$V = (U[L , *]^T \cdot U[L , *])^{-1} \cdot U[L , *]^T \cdot P[L , *] \quad (3.13)$$

Accuracy. Unlike proximity sketch, proximity embedding does not provide any provable data-independent accuracy guarantee. However, as a data-adaptive dimensionality reduction technique, when matrix P is in fact low-rank (*i.e.*, having good low-rank approximations), proximity embedding has the potential to achieve even better accuracy than proximity sketch. Our empirical results in Section 4.3 suggest that this is indeed the case for the Katz measure.

3.2.4 Evaluation

3.2.4.1 Dataset Description

We carry out our evaluation on five popular online social networks: Digg [28], Flickr [36], LiveJournal [61], MySpace [70], and YouTube [105]. For comparison, we also examine the hyperlink structure of Wikipedia [104].

Network	Snapshot Date	# of Connected Nodes	# of Links	# of Added Links	Asymmetric Link Fraction
Digg	9/15/2008	535,071	4,432,726	–	58.3%
	10/25/2008	567,771	4,813,668	656,478	
	11/10/2008	567,771	4,941,401	175,958	
Flickr	3/01/2007	1,932,735	26,702,209	–	37.8%
	4/15/2007	2,172,692	30,393,940	3,691,731	
	5/18/2007	2,172,692	32,399,243	2,005,303	
Live-Journal	11/13/2008	1,769,493	61,488,262	–	28.3%
	12/05/2008	1,769,543	61,921,736	1,566,059	
	1/30/2009	1,769,543	62,843,995	3,093,064	
MySpace	12/11/2008	2,128,945	89,138,628	–	0%
	1/11/2009	2,137,773	90,629,452	1,845,898	
	2/14/2009	2,137,773	89,341,780	696,016	
YouTube	4/30/2007	2,012,280	9,762,825	–	0%
	6/15/2007	2,532,050	13,017,064	3,254,239	
	7/23/2007	2,532,050	15,337,226	2,320,162	
Wikipedia	9/30/2006	1,636,961	28,950,137	–	83.1%
	12/31/2006	1,758,323	33,974,708	5,024,571	
	4/06/2007	1,758,323	38,349,329	4,374,621	

Table 3.1: Dataset summary

For each network, we conduct three crawls and make three snapshots of the network. Table 3.6 summarizes the characteristics of the three snapshots for each of the networks. Note that, for the purpose of link prediction, we only use connected nodes (*i.e.*, nodes with at least one incoming or outgoing friendship link), rather than considering all the crawled nodes. Another point to note is that since link prediction implies that based on one snapshot of the network, we predict the new links that are formed in the next snapshot, the same set of users should appear in two consecutive snapshots. Hence, for a growing network, the number of users appearing in the last snapshot that we create may be less than the total number of users (to match the previous snapshot). Lastly, although

there can be both link additions and deletions between two snapshots, since the goal of link prediction is to predict those that get added, we explicitly show the number of added links between two consecutive snapshots in Table 3.6.

Accuracy metrics. We quantify the estimation error using two different metrics: (i) *Normalized Mean Absolute Error* (NMAE) (defined as $\frac{\sum_i |est_i - actual_i|}{\sum_i actual_i}$), and (ii) *Relative Error* (defined as $\frac{|est_i - actual_i|}{actual_i}$), where est_i and $actual_i$ denote the estimated and actual values of the proximity measure for node pair i , respectively.

Since it is expensive to compute the actual proximity measures over all the data points, we randomly sample 100,000 data points by first randomly selecting 200 rows from the proximity matrix and then selecting 500 elements from each of these rows. We then compute errors for these 100,000 data points.

In the subsequent parts, we evaluate the accuracy and scalability of Proximity Embedding and Proximity Sketch using the aforementioned six datasets. We present results for Katz and RPR (defined in Section 3.1.1). The accuracy for escape probability (EP) is similar to RPR (due to their close relationship in Eq. 3.1) and is omitted in the interest of brevity.

3.2.4.2 Accuracy Evaluation of Proximity Embedding

We first evaluate the accuracy of Proximity Embedding in estimating Katz and RPR.

Parameter settings. Throughout our evaluation, we use a damping factor

of $\beta = 0.05$, $\ell_{\max} = 6$, and 1600 landmarks unless otherwise specified. We also vary these parameters to understand their impact. By default, we select landmarks based on the PageRank of each node. Specifically, we first compute PageRank for each node and normalize the sum of PageRank of all nodes to 1. We then use the normalized PageRank as the probability of assigning a node as a landmark. In this way, nodes with high PageRank values are more likely to become landmarks. For comparison, we also examine the performance of uniform landmark selection, which selects landmarks uniformly at random.

Estimating large Katz values. Table 3.2(a) shows the accuracy of proximity embedding in estimating Katz values larger than 1%, 0.1% and 0% of their corresponding row sums in the **Katz** matrix. As we can see, for elements larger than 0, the NMAE is low (the largest one is 0.0212 for YouTube). In comparison, for elements larger than 1% and 0.1% of the row sums, the NMAE is often larger (*e.g.*, the corresponding values for LiveJournal are 0.98 and 0.25). Manual inspection suggests that many Katz values larger than 1% of row sum involve a direct link between two nodes in an isolated island of the network that cannot reach any landmarks. For such node pairs, the proximity embedding yields an estimate of 0, thus seriously inflating the NMAE. Fortunately, large Katz values are quite rare in each row. As a result, the NMAE is low when we consider all elements in the Katz matrix.

Estimating large Rooted PageRank values. Table 3.3(a) shows the accuracy of proximity embedding in estimating Rooted PageRank values larger than 1%, 0.1%, and 0% of their corresponding row sums in the **RPR** matrix.

Network	Threshold for Large Katz Values		
	1% row sum	0.1% row sum	0% row sum
Digg	0.0562	0.0650	0.0002
Flickr	0.2177	0.2505	0.0001
LiveJournal	0.9872	0.2516	0.0122
MySpace	0.0532	0.0650	0.0002
YouTube	0.0074	0.0054	0.0212
Wikipedia	0.0039	0.0001	0.0027

(a) NMAE of proximity embedding

Network	Threshold for Large Katz Values		
	1% row sum	0.1% row sum	0% row sum
Digg	0.0001	0.3209	211.5
Flickr	0.0048	0.0293	1116.3
LiveJournal	0.0012	0.0113	1383.2
MySpace	0.0041	0.0360	1451.1
YouTube	0.0495	0.3769	1141.3
Wikipedia	0.0114	0.2645	647.3

(b) NMAE of proximity sketch

Table 3.2: Comparing proximity embedding and proximity sketch in estimating large Katz values.

We observe that the NMAE for RPR is much larger than the NMAE for Katz.

To understand why proximity embedding performs well on Katz but not on RPR, we plot the fraction of total variance captured by the best rank- k approximation to the inter-landmark proximity matrices $\mathbf{Katz}[L, L]$ and $\mathbf{RPR}[L, L]$ in Figure 3.3, where L is the set of landmarks. Note that the best rank- k approximation to a matrix can be easily computed through the use of singular value decomposition (SVD). The smaller the number of dimensions (*i.e.*, k) it takes to capture most variance of the matrix, the lower the “intrinsic” dimensionality the matrix has. As we can see, for LiveJournal, even 3 dimensions can capture over 99% variance for Katz, whereas it takes 1580

Network	Threshold for Large RPR Values		
	1% row sum	0.1% row sum	0% row sum
Digg	0.6662	2.0008	0.7933
Flickr	0.7285	1.6385	1.0000
LiveJournal	1.4491	7.2752	0.9980
MySpace	1.0916	6.6324	0.9984
YouTube	0.7068	1.1952	1.0635
Wikipedia	1.4429	5.7987	0.7208

(a) NMAE of proximity embedding

Network	Threshold for Large RPR Values		
	1% row sum	0.1% row sum	0% row sum
Digg	0.0031	0.0247	131.8
Flickr	0.0006	0.0019	717.6
LiveJournal	0.0042	0.0296	500.2
MySpace	0.0038	0.0269	853.0
YouTube	0.0019	0.0110	486.3
Wikipedia	0.0046	0.0265	619.7

(b) NMAE of proximity sketch

Table 3.3: Comparing proximity embedding and proximity sketch in estimating large RPR values.

dimensions to achieve similar approximation accuracy for rooted PageRank. This indicates that the RPR matrix is not low-rank, whereas the Katz matrix exhibits low-rank property, which makes proximity embedding work well.

Relative errors. Figure 3.4 further plots the CDF of relative errors using 60 dimensions. We take top 1%, 5%, and 10% of the randomly selected data points and generate the CDF for each of the selections. In all datasets, we observe that the relative errors are smaller for elements with larger values. This is desirable because larger elements play a more important role in many applications and are thus more important to estimate accurately.

3.2.4.3 Accuracy Evaluation of Proximity Sketch

Now we evaluate the accuracy of proximity sketch. We use $H = 3$ hash tables and $c = 1600$ columns in each table.

Estimating large Katz values. Table 3.2(b) shows the NMAE of proximity sketch in estimating Katz values larger than 1%, 0.1%, and 0% of row sum. Comparing Table 3.2(a) and (b), we observe that proximity sketch generally performs better than proximity embedding for large values (*i.e.*, greater than 1% of row sums), and performs worse for small values (*i.e.*, greater than 0.1% and 0 of row sums). This is consistent with our expectation, since proximity sketch is designed to approximate large elements. This also suggests that the two algorithms are complimentary and we can potentially have a hybrid algorithm that chooses the results among the two algorithms based on the magnitude of the estimated values.

Estimating large Rooted PageRank values. Table 3.3(b) shows the NMAE of proximity sketch in estimating RPR. As for Katz, proximity sketch yields lower error than proximity embedding for large elements (*i.e.*, larger than 1%, and 0.1% of row sums) and higher error for small elements (*i.e.*, larger than 0). This confirms that proximity sketch is effective in estimating large elements.

Dataset	proximity embedding			proximity sketch			Common neighbor		Shortest path distance	
Job type	Preprocess	Query		Preprocess	Query		Query		Query	
Sample type		Pos	Neg		Pos	Neg	Pos	Neg	Pos	Neg
Digg	1.08hrs	0.1 μ s	0.1 μ s	1.16hrs	0.3 μ s	0.2 μ s	15.0 μ s	12.0 μ s	149.2 μ s	132.5 μ s
Flickr	4.26hrs	47.9 μ s	11.8 μ s	4.88hrs	45.2 μ s	32.5 μ s	478.9 μ s	118.0 μ s	3111.1 μ s	1720.4 μ s
LiveJournal	8.29hrs	14.6 μ s	13.7 μ s	8.71hrs	15.1 μ s	13.4 μ s	546.2 μ s	137.5 μ s	9976.8 μ s	2416.7 μ s
MySpace	4.92hrs	58.8 μ s	84.1 μ s	9.85hrs	62.0 μ s	88.5 μ s	1588.1 μ s	841.8 μ s	50273.3 μ s	41473.2 μ s
YouTube	2.27hrs	25.1 μ s	20.6 μ s	4.67hrs	32.4 μ s	35.6 μ s	251.6 μ s	206.4 μ s	3727.5 μ s	1029.9 μ s
Wikipedia	3.30hrs	0.5 μ s	0.2 μ s	3.75hrs	0.6 μ s	0.2 μ s	54.0 μ s	24.0 μ s	1377.9 μ s	374.6 μ s

Table 3.4: Computation time of proximity embedding, proximity sketch, common neighbor and shortest path distance.

3.2.4.4 Scalability

Table 3.4 shows computation time for Proximity Embedding and Proximity Sketch compared with that of common neighbor (*e.g.*, # of common friends in the graph), and graph distance (*e.g.*, the shortest path distance). The measurements are taken on an Intel Core 2 Duo 2.33GHz machine with 4GB memory running Ubuntu Linux kernel v2.6.24. We explicitly distinguish the query time for positive and negative samples – A node pair $\langle A, B \rangle$ is considered a positive sample if there is a friendship link from A to B ; otherwise it is considered a negative sample.

We make several observations. First, the query time of both proximity embedding and proximity sketch is small, even smaller than common neighbor and graph distance, which are traditionally considered much cheaper operations than computing Katz. Second, the preprocessing of proximity embedding and proximity sketch increases linearly with the number of links in the dataset. As the preprocessing can be done in parallel, we use the Condor system [31] for datasets with a large number of links. Running preprocessing step simultaneously from 150 machines, we observe that the total preprocessing time goes down to less than 30 minutes, even for large networks such as

MySpace and LiveJournal. Furthermore, the pre-processing only needs to be done periodically (*e.g.*, once every few days). For symmetric networks, such as MySpace and YouTube, proximity embedding only needs to compute either $P[L, *]$ or $P[*, L]$, reducing the preparation time to half of that of proximity sketch. These results demonstrate the scalability of our approaches.

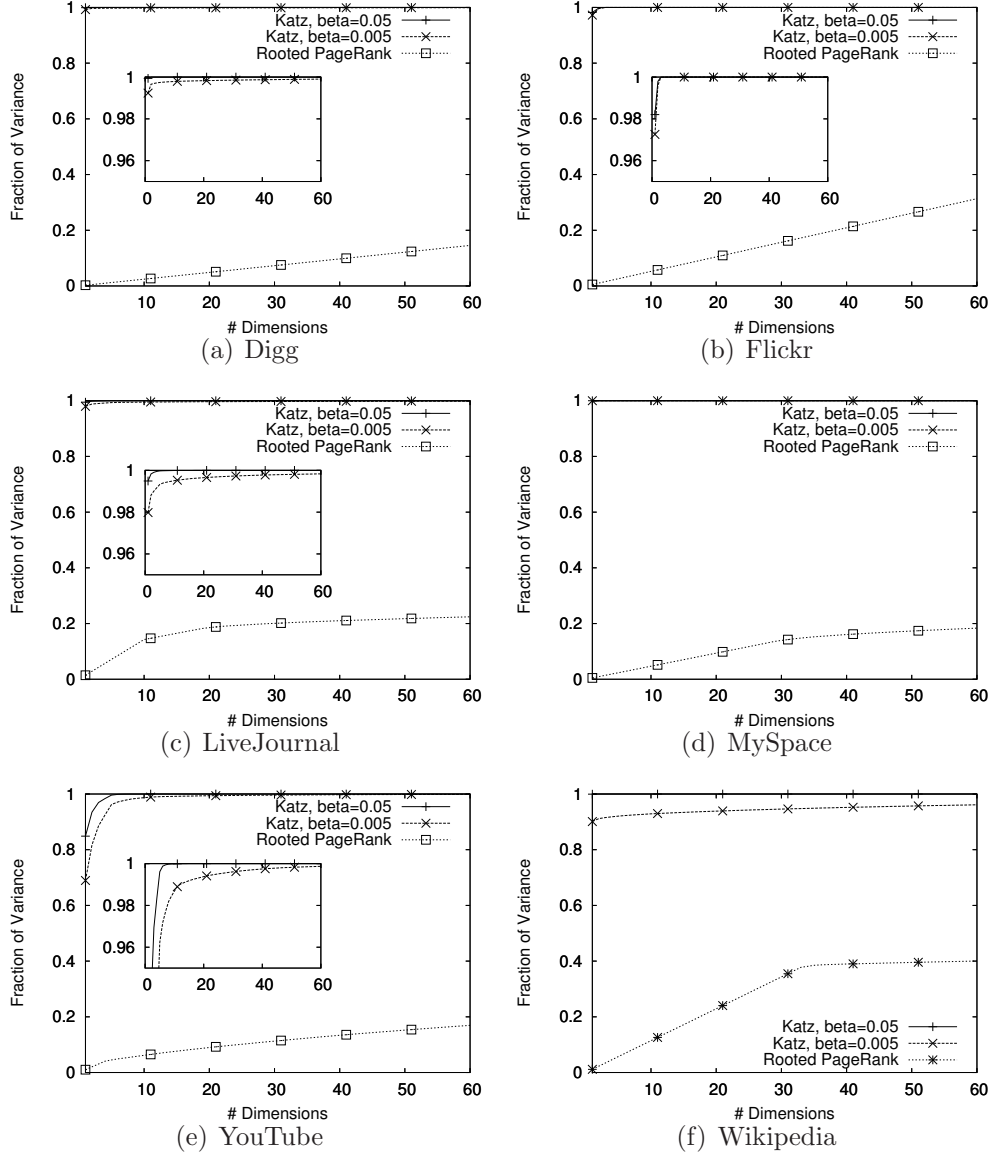


Figure 3.3: Fraction of total variance captured by the best rank- k approximation to inter-landmark proximity matrices $\text{Katz}[L, L]$ and $\text{RPR}[L, L]$.

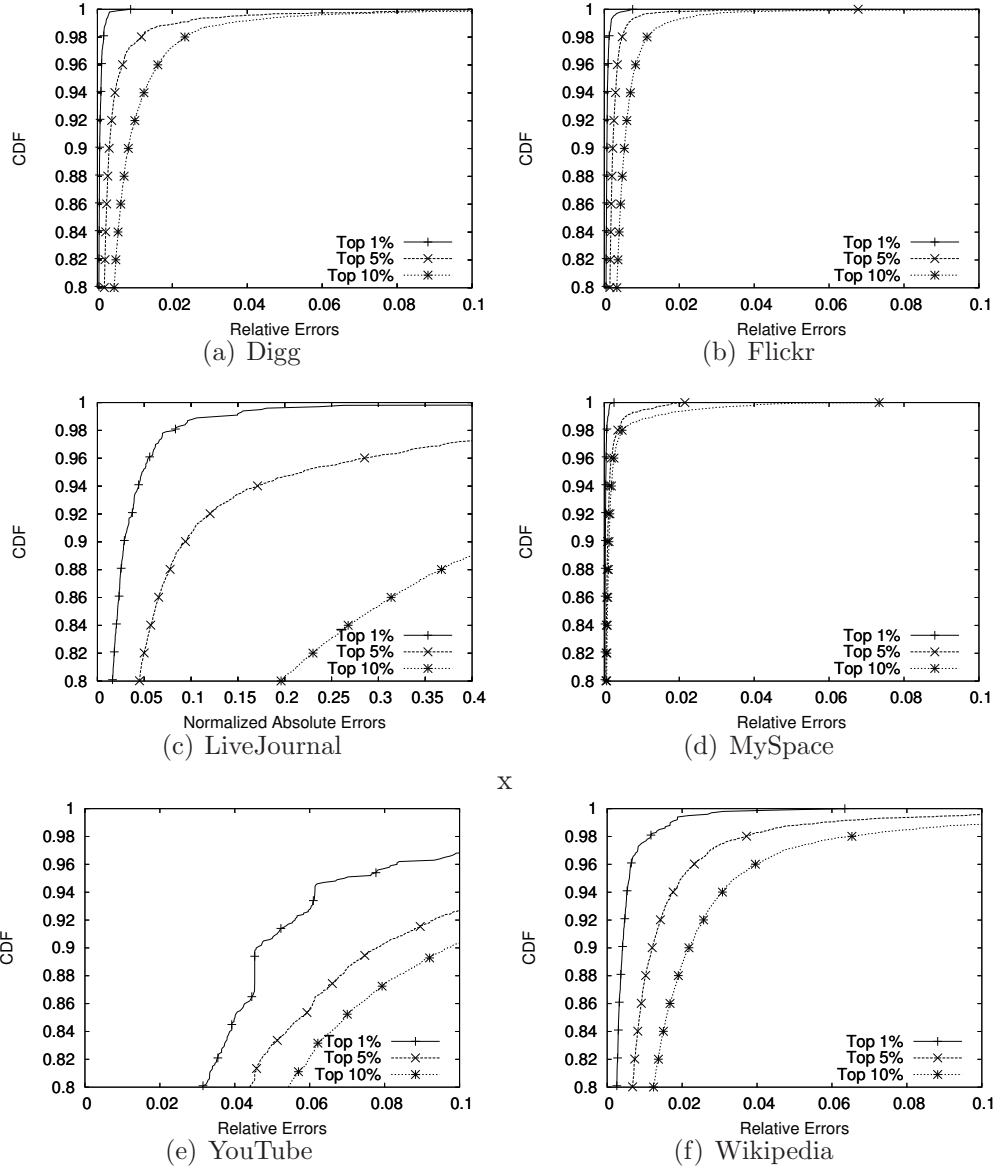


Figure 3.4: Relative errors for top 1%, 5%, and 10% largest values (Katz measure, $\beta = 0.05$, 1600 landmarks, and 60 dimensions).

3.3 Clustered Spectral Graph Embedding

3.3.1 Proposed Methodology

In this section, we describe our proposed *clustered spectral graph embedding* algorithm beginning by formalizing the following preliminary concepts.

Graph Embedding

Let A be an $m \times m$ adjacency matrix of a graph. For simplicity, we assume that A is symmetric. In section 3.3.1.3 we will show how to extend the presented concepts to non-symmetric adjacency matrices. A *graph embedding* can be mathematically formalized as the decomposition

$$A_{m \times m} \approx U_{m \times r} L_{r \times r} U_{m \times r}^T = ULU^T, \quad (3.14)$$

where U is an $m \times r$ orthonormal matrix (*i.e.*, $U^T U = I_r$ is an identity matrix), and L is an $r \times r$ matrix. U represents a basis for the *embedding subspace* and L represents the *embedded adjacency matrix* of the original graph. Since U is orthonormal, (Eq. 3.14) can be applied to approximate any matrix power A^k by

$$A^k \approx (ULU^T)^k = UL^k U^T. \quad (3.15)$$

As a special case, with $k = 2$, we get the frequently used common neighbor proximity measure $P_{\text{cn}} = A^2 \approx UL^2 U^T$. Many functions defined on A can be approximated using a sum of matrix powers through the Taylor series expansion. Using (Eq. 3.15) we can approximate these functions with corresponding functions on L . This can significantly reduce the computational cost because

$r \ll m$ and most of the calculations will involve the $r \times r$ matrix L instead of the $m \times m$ matrix A . For example, using (Eq. 3.14) and (Eq. 3.15), we can approximate the Katz measure as

$$P_{\text{Kz}} \approx \sum_{k=1}^{\infty} \beta^k U L^k U^{\text{T}} = U \left((I_r - \beta L)^{-1} - I_r \right) U^{\text{T}}.$$

Spectral graph embedding

The best rank- r approximation of A is given by the r -dimensional *spectral graph embedding* (SGE)

$$A \approx U \Lambda U^{\text{T}}, \quad (3.16)$$

where Λ is a diagonal matrix with the r largest (in magnitude) eigenvalues of A , and U contains the corresponding eigenvectors. Figure 3.6(a) shows a pictorial illustration of the spectral graph embedding.

Over the past several decades, eigen decomposition and spectral graph embedding have been the primary research tool for achieving dimensionality reduction on large matrices and graphs. Although there are computationally efficient algorithms [52, 51] to compute the spectral embedding of large sparse matrices, they are still quite expensive on massive social networks with million of nodes. As a result, spectral graph embedding can only afford to work with relatively small r , which may not capture sufficient social/network structure for very large social networks and thus yields poor approximation accuracy (see Section 4.3).

Graph clustering

A key step in the methods we will propose is to cluster or partition a graph. Given a graph $G = (\mathcal{V}, \mathcal{E})$ there are various objective functions that measure the quality of the clustering, *e.g.*, *ratio cut* [43] and *normalized cut* [86]. Although the graph clustering objectives are NP-hard problems [101], there are several efficient clustering algorithms that often produce good quality partitioning of a graph, *e.g.*, Graclus [27], METIS [2], “modularity” optimization [10], and power iteration clustering [59].

Assume that we have a clustering of $G(\mathcal{V}, \mathcal{E})$ into c disjoint clusters specified by the vertex sets \mathcal{V}_i , $i = 1, \dots, c$, *i.e.*, $\bigcup_{i=1}^c \mathcal{V}_i = \mathcal{V}$ and $\mathcal{V}_i \cap \mathcal{V}_j = \emptyset$ for all $i \neq j$. Let $m_i = |\mathcal{V}_i|$. Without loss of generality, we can assume that the vertices in $\mathcal{V}_1, \dots, \mathcal{V}_c$ are in a strictly increasing order. Then the adjacency matrix A will have the following form

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1c} \\ \vdots & \ddots & \vdots \\ A_{c1} & \cdots & A_{cc} \end{bmatrix}, \quad (3.17)$$

where each diagonal block A_{ii} is an $m_i \times m_i$ matrix, that can be considered as a local adjacency matrix for cluster i . The off-diagonal blocks A_{ij} ($i \neq j$) are $m_i \times m_j$ matrices that contain the set of edges incident on vertices belonging to different clusters. In an ideal scenario, with perfect clustering, the off-diagonal blocks will not contain any edges, thus $A_{ij} = 0$, and the graph will have c connected components. On the other hand, in the more general scenario of A comprised of a single connected component with a clear clustering structure, a naive approximation of the original graph could be obtained through its diagonal blocks $A \approx \text{diag}(A_{11}, \dots, A_{cc})$. By further introducing

low-rank approximations $A_{ii} \approx V_i \Lambda_i V_i^\top$, we have

$$A \approx \text{diag}(V_1 \Lambda_1 V_1^\top, \dots, V_c \Lambda_c V_c^\top).$$

3.3.1.1 Basic algorithm

Our proposed method, clustered spectral graph embedding improves the efficiency and accuracy of approximating various proximity measures by effectively combining clustering with spectral graph embedding. Recall that A is the adjacency matrix of a graph. We will assume that the graph has been partitioned into c clusters and that the vertices are ordered so that the diagonal blocks A_{ii} , $i = 1, \dots, c$, correspond to the local adjacency matrices for the different clusters as in (Eq. 3.17). Since the block partitioning of A is obtained through clustering of the graph, it follows that most of the edges are within the diagonal blocks. Only a small fraction of the edges are between clusters and are consequently located in the off-diagonal blocks. Figure 3.5 shows a typical sparsity pattern of an adjacency matrix after a clustering step. In this particular case, we have clustered the graph into $c = 10$ clusters and 80% of the edges are within the diagonal blocks. Computing the best rank- r_i approximations through the spectral graph embedding for every cluster (diagonal block), we get

$$A_{ii} \approx V_i \Lambda_i V_i^\top, \quad i = 1, \dots, c, \quad (3.18)$$

where Λ_i is a diagonal matrix and contains the r_i largest eigenvalues of A_{ii} (in magnitude), and V_i is an orthonormal matrix with the corresponding eigen-

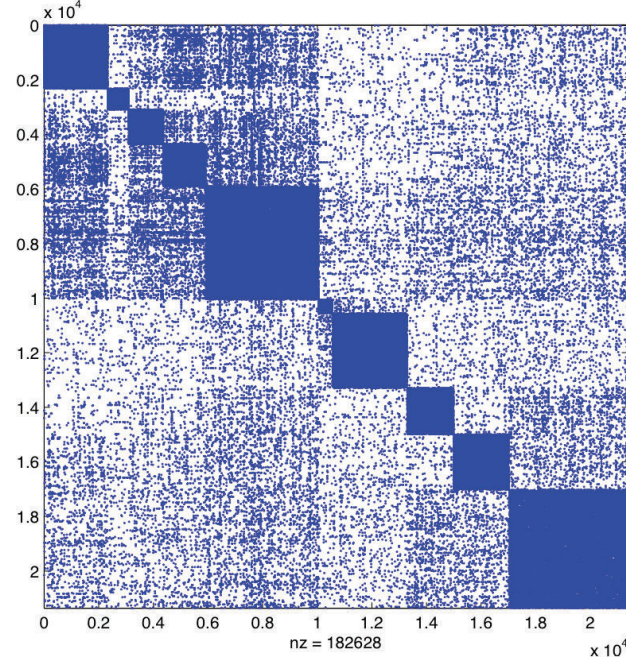


Figure 3.5: Sparsity pattern of an adjacency matrix with $c = 10$ clusters. 80% edges are within diagonal blocks.

vectors. Due to the orthonormality of V_i it follows that the matrix

$$V = \text{diag}(V_1, \dots, V_c)$$

is also orthonormal. We can now use this block-diagonal matrix V to obtain a graph embedding for the entire adjacency matrix A . The graph embedding may be written as $A \approx VSV^\top$. Since V is orthonormal, it follows that the optimal S , in least squares sense, is

$$S = V^\top AV = \begin{bmatrix} S_{11} & \cdots & S_{1c} \\ \vdots & \ddots & \vdots \\ S_{c1} & \cdots & S_{cc} \end{bmatrix},$$

where $S_{ij} = V_i^\top A_{ij} V_j$, for $i, j = 1, \dots, c$. Using (Eq. 3.18) we can verify that $S_{ii} = \Lambda_i$ are diagonal. The off-diagonal blocks S_{ij} , on the other hand, capture

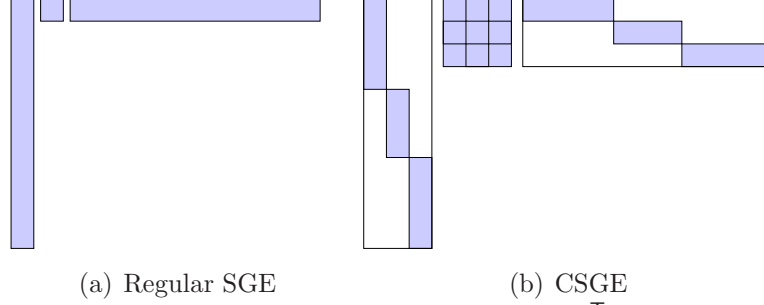


Figure 3.6: Illustration of the regular SGE $A \approx U\Lambda U^\top$ and CSGE $A \approx VSV^\top$ in (Eq. 3.19) with $c = 3$ clusters.

interactions between different clusters. We obtain the following approximation

$$A \approx VSV^\top = \text{diag}(V_1, \dots, V_c) \begin{bmatrix} S_{11} & \cdots & S_{1c} \\ \vdots & \ddots & \vdots \\ S_{c1} & \cdots & S_{cc} \end{bmatrix} \text{diag}(V_1, \dots, V_c)^\top, \quad (3.19)$$

which we denote as the *clustered spectral graph embedding* (CSGE). For example, with $c = 3$ we can write

$$A \approx \begin{bmatrix} V_1 & 0 & 0 \\ 0 & V_2 & 0 \\ 0 & 0 & V_3 \end{bmatrix} \begin{bmatrix} S_{11} & S_{12} & S_{13} \\ S_{21} & S_{22} & S_{23} \\ S_{31} & S_{32} & S_{33} \end{bmatrix} \begin{bmatrix} V_1 & 0 & 0 \\ 0 & V_2 & 0 \\ 0 & 0 & V_3 \end{bmatrix}^\top.$$

Figure 3.6 shows an illustration of the SGE compared to the CSGE. In analogy to the terminology for spectral graph embedding, we will denote $V = \text{diag}(V_1, \dots, V_c)$ as a basis for the *clustered embedding subspace*, and S as the *embedded adjacency matrix* for CSGE. Note that with $c = 1$, CSGE becomes the regular SGE.

3.3.1.2 Advantages

Compared with the rank- r SGE $A \approx U\Lambda U^\top$, the CSGE $A \approx VSV^\top$ from (Eq. 3.19) can achieve much higher accuracy while using a comparable

amount of memory. CSGE is also computationally more efficient. To simplify the discussion for the CSGE we will use r -dimensional graph embeddings for each cluster A_{ii} (*i.e.*, $r_i = r$).

Memory usage of the final embedding.

From the block-diagonal structure of V , we immediately observe that we have a cr -dimensional embedding in the CSGE, while only an r -dimensional embedding in the SGE. Thus, there are c times more columns in $V_{m \times cr}$ than in $U_{m \times r}$. However, $V_{m \times cr}$ has exactly the same $O(mr)$ memory usage as $U_{m \times r}$, since we only store the diagonal blocks V_i , and not the off-diagonal blocks, which are all zeros (see Figure 3.6). Comparing the embedded adjacency matrices S from the CSGE and Λ from the SGE, we see that the $cr \times cr$ matrix S is larger and dense, while the Λ is $r \times r$ and diagonal. Therefore, CSGE uses $O(mr + c^2r^2)$ total memory and SGE uses $O(mr + r)$. For large graphs with millions of vertices, the memory complexity is dominated by U or V because we have $m \gg r$ and $m \gg cr$. For example, in the LiveJournal dataset $m = 1,770,961$. With (typical values) $r = 100$ and $c = 50$, S accounts for only 12% of the total memory usage of CSGE. As shown in Section 3.3.1.3, we can further reduce the memory usage of S through another embedding. So the total memory usage of CSGE is comparable to that of SGE.

Memory usage for deriving the embedding.

With state-of-the-art algorithms for computing a small number of eigenvectors of a large and sparse matrix (*e.g.*, [52, 53]), the SGE has a memory

complexity of $O(m(r + p))$, where p is a user specified parameter and usually $p \approx r$. Smaller p requires less memory but the convergence becomes slower, and vice versa. For the CSGE, the approximation of the different clusters is independent of each other. These computations can be done using the entire memory available. Since the size of each cluster $m_i = |\mathcal{V}_i|$ may be orders of magnitude smaller than $m = \sum_{i=1}^c m_i$, we can compute the per-cluster spectral graph embedding with a much larger dimension than otherwise possible. For example, when $c = 10$, the average cluster size would be about $m/10$. Allowing the CSGE to fully utilize the entire memory space would increase the maximum computable dimensions from r to $10r$.

Computational efficiency.

The time complexity for computing the CSGE or the SGE is dominated by the cost of eigendecomposition. State-of-the-art algorithms (*e.g.*, [52, 53]) for sparse eigendecomposition are typically iterative algorithms. The time complexity for each iteration scales linearly with respect to the number of non-zeros in the input matrix, whereas the number of iterations required to achieve convergence depends on the gap between adjacent eigenvalues. For example, it typically takes many more iterations to decompose the normalized adjacency matrix $T = D^{-1/2}AD^{-1/2}$ than the original adjacency matrix A because eigenvalues of T are more evenly distributed.

It is easy to see that the number of non-zeros in the global adjacency matrix A is larger than the total number of non-zeros in all the per-cluster adjacency matrices A_{ii} . Hence, the per-iteration cost for decomposing A is

higher than the total per-iteration cost for decomposing all the A_{ii} . In addition, our experience suggests that it often takes a larger number of iterations for the global eigendecomposition to converge. As a result, it is often much faster to compute many r -dimensional graph embeddings of smaller matrices A_{ii} than a single r -dimensional graph embedding computation of a large A . Our experimental results in Section 4.3 show that even after including the initial clustering time of the graph as well as the computation time for S , the CSGE is still much faster than the SGE. In the specific case when the normalized adjacency matrix T is used, CSGE is an order of magnitude faster than SGE. Moreover, with parallel/distributed computation of clusters detailed in Section 3.3.1.3, we can further improve the timing efficiency of CSGE.

Accuracy

An important consequence of explicitly focusing on each cluster of the network is that using the same amount of memory the CSGE yields a significantly smaller residual than the residual for the regular SGE, *i.e.*,

$$\|A - VSV^T\|_F < \|A - U\Lambda U^T\|_F,$$

where $\|X\|_F = \sqrt{\sum_{ij} X[i, j]^2}$ is the Frobenius norm of X . Recall that the SGE is optimal with respect to the dimension of the embedding (or with respect to the rank in the approximation), but in terms of memory consumption the CSGE gives significantly better embedding (see Section 4.3).

The accuracy benefit of CSGE is most significant when we are able to transform (using permutations) a given adjacency matrix so that most of the

non-zeros are contained within the diagonal blocks. This property is closely related to a graph forming good clusters. Many (if not most) real-world graphs and social networks exhibit this property of forming reasonably good clusters [54, 55]. This is certainly the case for the datasets we have considered in this work.

Interestingly, even if the original graph does not form good clusters or if the clustering algorithm performs poorly, clustered graph embedding can still achieve lower residual error than the regular SGE. The following theorem establishes this guarantee for the special case with $c = 2$ clusters.

Theorem 2. *Let $A \approx U\Lambda U^\top$ be the r -dimensional SGE. Split U into any two parts $U^\top = [U_1^\top \ U_2^\top]$. Let $U_i = Q_i R_i$ be the QR decomposition [41] of U_i , where Q_i is orthonormal (i.e., $Q_i^\top Q_i = I_r$) and R_i is upper triangular. Let $V = \text{diag}(Q_1, Q_2)$ and let $S = V^\top A V$. We have*

$$\|A - V S V^\top\|_F \leq \|A - U \Lambda U^\top\|_F.$$

Proof. Through simple matrix calculation, we have:

$$\begin{aligned} U \Lambda U^\top &= \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix} \begin{bmatrix} \Lambda & \Lambda \\ \Lambda & \Lambda \end{bmatrix} \begin{bmatrix} U_1 & 0 \\ 0 & U_2 \end{bmatrix}^\top = \\ &\begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix} \begin{bmatrix} R_1 \Lambda R_1^\top & R_1 \Lambda R_2^\top \\ R_2 \Lambda R_1^\top & R_2 \Lambda R_2^\top \end{bmatrix} \begin{bmatrix} Q_1 & 0 \\ 0 & Q_2 \end{bmatrix}^\top \triangleq V \bar{\Lambda} V^\top. \end{aligned}$$

It is evident that V is orthonormal. So $S = V^\top A V$ is an optimal solution to $\min_S \|A - V S V^\top\|_F$. As a result, we have $\|A - V S V^\top\|_F \leq \|A - V \bar{\Lambda} V^\top\|_F = \|A - U \Lambda U^\top\|_F$. \square

The above theorem can be easily generalized to the case with $c > 2$ clusters. Therefore, under any arbitrary clustering of the original graph, there always exists a clustered graph embedding that has lower residual error than the regular SGE.

3.3.1.3 Extensions

The basic CSGE algorithm can be further extended to support (i) asymmetric adjacency matrices, (ii) further embedding of the embedded adjacency matrix, (iii) incremental updates, and (iv) parallel/distributed computation.

Asymmetric adjacency matrices. There are two possible strategies to cope with an asymmetric adjacency matrix A .

- When the fraction of asymmetric edges (*i.e.*, vertex pairs (i, j) such that $A[i, j] \neq A[j, i]$) is not high, we can simply derive the basis of the clustered embedding subspace (*i.e.*, the V matrix) using a symmetrized version of A , *e.g.*, $A_{\text{sym}} = \frac{1}{2}(A + A^T)$ or $A_{\text{sym}} = \max(A, A^T)$. We can then capture the asymmetry of A by solving $\min_S \|A - VSV^T\|_F$, yielding $S = V^T AV$.
- Alternatively, we can apply singular value decomposition [41] to compute the best rank- r_i approximation for every cluster (diagonal block), yielding

$$A_{ii} \approx U_i \Sigma_i V_i^T, \quad i = 1, \dots, c,$$

where Σ_i is a diagonal matrix containing the r_i largest singular values of A_{ii} , U_i and V_i are orthonormal matrices with the corresponding left and right singular vectors, respectively. Due to the orthonormality of U_i and V_i ,

$U \triangleq \text{diag}(U_1, \dots, U_c)$ and $V \triangleq \text{diag}(V_1, \dots, V_c)$ are also orthonormal. We can then use the block-diagonal matrices U and V to obtain an asymmetric clustered graph embedding for the entire adjacency matrix A written as $A \approx USV^\top$. The optimal S , in least squares sense, is

$$S = U^\top AV = \begin{bmatrix} S_{11} & \cdots & S_{1c} \\ \vdots & \ddots & \vdots \\ S_{c1} & \cdots & S_{cc} \end{bmatrix},$$

where $S_{ij} = U_i^\top A_{ij} V_j$, for $i, j = 1, \dots, c$. The asymmetric graph embedding $A \approx USV^\top$ has properties very similar to the symmetric graph embedding. For example, it can be applied to efficiently approximate matrix powers $A^k \approx (USV^\top)^k = U(SV^\top U)^{k-1} SV^\top$.

Further embedding. With a large number of clusters, and larger ranks in the approximations, the size of S could become too large. To reduce memory usage, we can further compute the SGE of the embedded adjacency matrix S . That is, we further approximate S by keeping the largest eigenvalues (in magnitude), *i.e.*, $S \approx \bar{Q} \bar{\Lambda} \bar{Q}^\top$, where $\bar{\Lambda}$ contains the largest eigenvalues of S and \bar{Q} contains the corresponding eigenvectors. The combined approximation is then $A \approx V \bar{Q} \bar{\Lambda} \bar{Q}^\top V^\top$. Note that we can directly derive the SGE $S \approx \bar{Q} \bar{\Lambda} \bar{Q}^\top$ without having to first compute the dense matrix S (which requires extra computation and memory). All we need is to treat $S = V^\top AV$ as an operator acting on a vector. Thus, for a given vector v we compute $Sv = (V^\top AV)v = V^\top(A(Vv))$. Since V is block diagonal, and A is sparse, computing matrix-vector products with both V and A is efficient. State-of-the-art algorithms

for sparse eigendecomposition (*e.g.*, [52, 53]) can then efficiently compute the SGE $S \approx \bar{Q}\bar{\Lambda}\bar{Q}^\top$ using such matrix-vector products.

Incremental updates. As many social networks are highly dynamic, it is desirable to cheaply update the graph embedding equation $A \approx VSV^\top$ under the new adjacency matrix $A' = A + \Delta_A$ when Δ_A has only few non-zero elements. A simple strategy is to keep V stable while updating S with

$$\begin{aligned} S' &= V^\top A' V = V^\top (A + \Delta_A) V \\ &= S + V^\top \Delta_A V \triangleq S + \Delta_S, \end{aligned} \tag{3.20}$$

where $\Delta_S = V^\top \Delta_A V$ contains the updates to S and can be efficiently computed due to the sparsity of Δ_A . The embedding for the new adjacency matrix A' is then $A' \approx VS'V^\top$.

Parallel/distributed computation. CSGE is naturally suited for parallel/distributed computation. For example, consider the email exchange graph of a large corporation, which may consist of a number of organizations. Instead of requiring a separate clustering step, we can directly partition users based on their natural organizational boundaries. Each organization (*i.e.*, cluster) i can then derive V_i locally based on its internal email exchange subgraph A_{ii} according to $A_{ii} \approx V_i \Lambda_i V_i^\top$. Each pair of organizations i and j can jointly compute the block S_{ij} of the embedded adjacency matrix S based on their V_i , V_j and the inter-organization email exchange subgraph A_{ij} , which can again be locally monitored. The ability to support parallel/distributed computation

can further improve efficiency in parallel/distributed computational environments.

3.3.2 Applications

In this section, we show how to utilize CSGE and SGE in three important social network analysis tasks: *proximity estimation*, *missing link inference*, and *link prediction*.

3.3.2.1 Proximity estimation

In Section 3.1.1 we defined four proximity measures: common neighbors, Katz measure, rooted PageRank, and escape probability. Each of these proximity measures will be computed or approximated using three different approaches: (i) direct computation; (ii) approximation using the spectral graph embedding $A \approx U\Lambda U^\top$ from (Eq. 3.16); and (iii) approximation using the clustered spectral graph embedding $A \approx VSV^\top$ from (Eq. 3.19). We obtain 12 different proximity measures as summarized in Table 3.5. Note that following [88], the summation for the Katz measures are truncated to only include $k_{\max} = 6$ terms, *i.e.*, at most length- k_{\max} paths are taken into account. The truncation is necessary in order to compute the real proximity measures that do not use any graph embedding [88], which is used as the ground truth for accuracy evaluation. In the rooted PageRank measures, we truncate the summation to only include $k_{\max} = 20$ terms and we have the spectral and the clustered spectral graph embeddings $T \triangleq D^{-1/2}AD^{-1/2} \approx \bar{U}\bar{\Lambda}\bar{U}^\top$ and $T \approx \bar{V}\bar{S}\bar{V}^\top$,

respectively. Finally, the escape probability measures are computed from the rooted PageRank measures using the function f defined in (Eq. 3.1).

Abbreviation	Method name	Definition
cn	Common neighbor	$P_{cn} = A^2$
cn-sge	Common neighbor with SGE	$P_{cn-sge} = U \Lambda^2 U^\top$
cn-csge	Common neighbor with CSGE	$P_{cn-csge} = V S^2 V^\top$
kz	Katz measure	$P_{kz} = \sum_{k=1}^{k_{\max}} \beta^k A^k$
kz-sge	Katz measure with SGE	$P_{kz-sge} = U (\sum_{k=1}^{k_{\max}} \beta^k \Lambda^k) U^\top$
kz-csge	Katz measure with CSGE	$P_{kz-csge} = V (\sum_{k=1}^{k_{\max}} \beta^k S^k) V^\top$
rpr	Rooted PageRank	$P_{rpr} = (1 - \alpha) D^{-1/2} (\sum_{k=0}^{k_{\max}} \alpha^k T^k) D^{1/2}$
rpr-sge	Rooted PageRank with SGE	$P_{rpr-sge} = (1 - \alpha) D^{-1/2} (I + \bar{U} (\sum_{k=1}^{k_{\max}} \alpha^k \bar{\Lambda}^k) \bar{U}^\top) D^{1/2}$
rpr-csge	Rooted PageRank with CSGE	$P_{rpr-csge} = (1 - \alpha) D^{-1/2} (I + \bar{V} (\sum_{k=1}^{k_{\max}} \alpha^k \bar{S}^k) \bar{V}^\top) D^{1/2}$
ep	Escape probability	$P_{ep}[i, j] = f(P_{rpr}, i, j)$
ep-sge	Escape probability with SGE	$P_{ep-sge}[i, j] = f(P_{rpr-sge}, i, j)$
ep-csge	Escape probability with CSGE	$P_{ep-csge}[i, j] = f(P_{rpr-csge}, i, j)$

Table 3.5: Description of the proximity measures.

3.3.2.2 Missing link inference

Missing link inference aims to infer additional links that, while not directly visible, are likely to exist (based on the set of links that are directly observed). Missing link inference falls into the general realm of compressive sensing, which aims to reconstruct missing data based on indirect, partial observations. Compressive sensing has many applications both in network-ing [111] and beyond, and has attracted considerable research attention recently (*e.g.*, [30, 82, 12, 111]).

Problem definition. Let $G = (\mathcal{V}, \mathcal{E})$ be a social network with a binary adjacency matrix A . Let \mathcal{O} be the set of observed edges or links and $\mathcal{M} = \mathcal{E} \setminus \mathcal{O}$ be the set of missing or unobserved links. Let $A_{\mathcal{O}}$ be the incomplete adjacency matrix containing the set of observed links. $A_{\mathcal{O}}$ is defined as

$$A_{\mathcal{O}}[i, j] = \begin{cases} A[i, j], & \text{if } (i, j) \in \mathcal{O}; \\ 0, & \text{otherwise.} \end{cases}$$

Let $A_{\mathcal{M}} = A - A_{\mathcal{O}}$. Given $A_{\mathcal{O}}$, the goal of missing link inference is to infer non-zeros in $A_{\mathcal{M}}$ as they correspond to the set of missing links in \mathcal{M} .

Inference algorithm. Despite much progress on compressive sensing, we are not aware of any existing compressive sensing algorithm that can scale to massive social networks with millions of vertices. Therefore, in this work we explore the following simple but much more scalable heuristic. We first compute proximity measures based on the observed incomplete adjacency matrix $A_{\mathcal{O}}$ (*e.g.*, those given in Section 3.3.2.1), and then assume that the high values in these proximity measures will correspond to the set of missing links, *i.e.*, non-zeros in $A_{\mathcal{M}}$. The threshold for determining high proximity measure values can be varied to achieve different tradeoffs between false positives and false negatives.

3.3.2.3 Link prediction

Link prediction [57] refers to the task of predicting which vertex pairs in a social network will become connected. An accurate link predictor can provide valuable insights for constructing meaningful network evolution models [8, 24]. It also allows social networks to make high-quality recommendations on potential new friends, making it easier for individual users to expand their social neighborhood. Link prediction also has direct applications in cyber security. For example, it allows one to conjecture that certain individuals in a terrorist network are working together even though their interaction has not been directly observed [50]. In this section, we show how CSGE and supervised

learning together can facilitate more accurate link prediction.

Problem specification

A natural setting for evolving social networks, or evolving networks in general, is to introduce discrete time steps $t = 1, \dots, t_{\max}$ at which a “snapshot” $G_t = (\mathcal{V}_t, \mathcal{E}_t)$ of the graph is taken. Denote the corresponding adjacency matrices with A_t , $t = 1, \dots, t_{\max}$. To ease the analysis and notation, we will restrict ourselves and only use \mathcal{V}_1 for all time steps, *i.e.*, $\mathcal{V}_k = \mathcal{V}_1 \triangleq \mathcal{V}$. It is clear that the evolution of the graph is incremental. In terms of the adjacency matrices, we can express this as $A_{t+1} = A_t + \Delta_t$, where Δ_t contains the edges or links that are formed between time t and $t + 1$ (for simplicity, we assume no edges are removed).

In the link prediction problem, given A_t , we try to find the non-zeros in Δ_t as they correspond to newly formed links. The standard heuristic is to first compute some proximity measures based on A_t (*e.g.*, those given in Section 3.3.2.1) and then assume that the high scores in these proximity measures will correspond to new links, *i.e.*, non-zeros in Δ_t . In the following, we will present several supervised proximity measures that explicitly target Δ_t and construct graph specific models for link prediction. To clarify the problem setting, we will use three snapshots, A_1 , A_2 , and A_3 . The first two will be used to learn the model and the third one will be used to validate the obtained model. Extending to more than three snapshots is straightforward.

Link prediction models. The link prediction models we propose have the

generic prediction scores, obtained from a low dimensional graph embedding

$$P_* = W_* F_*(x) W_*^\top,$$

where W_* represents the basis for a graph embedding, and $F_*(x)$ is a small matrix with the model parameters x that will be learned. In particular, we will consider the following two models that are characterized by the particular form of the embedded adjacency matrix $F_*(x)$ and the kind of graph embedding (spectral or clustered spectral) that is used.

- *Spectral learning.* In this model, we set $W_{\text{sl-sge}} = U_t$, where $A_t \approx U_t \Lambda_t U_t^\top$ is the SGE of the adjacency matrix A_t . We then let the parameter matrix be of the form

$$F_{\text{sl-sge}}(x) = \text{diag}(x_1, \dots, x_r).$$

- *Clustered spectral learning.* Here we set $W_{\text{sl-csge}} = V_t$, where $A_t \approx V_t S_t V_t^\top$ is the CSGE of the adjacency matrix A_t as in (Eq. 3.19). The parameter matrix has the form

$$F_{\text{sl-csge}}(x) = Q_t \text{diag}(x_1, \dots, x_{cr}) Q_t^\top,$$

where orthogonal matrix Q_t is obtained from the full eigendecomposition $S_t = Q_t \Lambda_{S_t} Q_t^\top$. Recall that V_t is block diagonal and S_t is a dense matrix that captures interactions between all clusters. The number of parameters to be learned is cr , where c is the number of clusters and r is the embedding dimension for each.

Framework. Our basic framework for supervised link prediction consists of the following two phases:

1. *Supervised learning.* We compute the embedded adjacency matrix W_* from either the SGE or the CSGE of the first snapshot A_1 . We then learn the parameter matrix $F_*(x)$ by performing linear regression on $\Delta_1 = A_2 - A_1$ (which contains the links newly formed between snapshots A_1 and A_2) with respect to $F_*(x)$.
2. *Link prediction.* We compute the embedded adjacency matrix \bar{W}_* from either the SGE or the CSGE of the second snapshot A_2 . We then use $P_* = \bar{W}_* F_*(x) \bar{W}_*^T$ as the prediction score matrix to predict $\Delta_2 = A_3 - A_2$, which contains the set of new links that will be formed between snapshots A_2 and A_3 .

3.3.2.4 Supervised learning

Supervised learning. At each time step t , there are three associated sets: the set of existing edges \mathcal{E}_t ; a positive set \mathcal{P}_t containing vertex pairs that form new links in the time interval $(t, t+1]$; and a negative set \mathcal{N}_t containing vertex pairs without edges at time step $t+1$. Using adjacency matrices A_t and A_{t+1} , we can write

$$\mathcal{E}_t = \{(i, j) \mid A_t[i, j] \neq 0\},$$

$$\mathcal{P}_t = \{(i, j) \mid A_t[i, j] = 0 \text{ and } A_{t+1}[i, j] \neq 0\},$$

$$\mathcal{N}_t = \{(i, j) \mid A_t[i, j] = 0 \text{ and } A_{t+1}[i, j] = 0\}.$$

It is straightforward to see that $\mathcal{E}_{t+1} = \mathcal{E}_t \cup \mathcal{P}_t$, and that the three sets \mathcal{E}_t , \mathcal{P}_t , \mathcal{N}_t are mutually disjoint.

Supervised learning of model parameters. We perform linear regression on Δ_1 to learn the parameter matrix $F_*(x)$. Specifically, we solve the following least squares problem:

$$\min_x \sum_{(i,j) \in \mathcal{S}} ([W_* F_*(x) W_*^\top]_{i,j} - \Delta_1[i, j])^2, \quad (3.21)$$

where W_* is obtained from either the SGE or the CSGE of A_1 , $F_*(x)$ is one of the parameter matrices from Section 3.3.2.3 (for time step $t = 1$), $\Delta_1 = A_2 - A_1$ contains the new links, the notation $[\cdot]_{i,j}$ denotes the i, j entry of the argument, and \mathcal{S} is the sample set (see discussions below). Care must be taken in order to solve this least squares problem efficiently. In the interest of brevity, we omit the details of the solution process and will include them in a technical report.

Choice of the sample set \mathcal{S} . In the ideal case the model parameters in the vector x should be learned only over the positive set \mathcal{P}_1 and the negative set \mathcal{N}_1 , thus setting $\mathcal{S} = \mathcal{P}_1 \cup \mathcal{N}_1$. With this approach, the edges that already exist in \mathcal{E}_1 , *i.e.*, the non-zeros in A_1 , will not contribute to the learning process. The reason behind this choice of the sample set is that we only want to target links that will form during the next time interval and we do not want to target links that already exist. Unfortunately, for the social networks we have considered, the choice $\mathcal{S} = \mathcal{P}_1 \cup \mathcal{N}_1$ would yield a sample set of the order $|\mathcal{V}|^2$ and practically impossible to work with. To make the problem manageable,

we choose \mathcal{S} to contain a fraction of \mathcal{P}_1 and a fraction of \mathcal{N}_1 . In addition $|\mathcal{S}|$ should be not only large enough to capture the essence of the model, but also have manageable size. In our experiments we have $|\mathcal{V}| \approx 2 \cdot 10^6$ and we choose $|\mathcal{S}| \approx 5 \cdot 10^5$ (see Section 4.3).

Link prediction The different models are validated by predicting the new links in A_3 . The prediction scores are based on

$$P_* = \bar{W}_* F_* \bar{W}_*^T,$$

where \bar{W}_* is obtained from a SGE or CSGE based on A_2 (instead of A_1 used in the model learning step). Specifically, let the SGE of A_2 be $A_2 \approx U_2 \Lambda_2 U_2^T$, then the prediction scores for **sl-sge** can be written as

$$P_{\text{sl-sge}} = U_2 F_{\text{sl-sge}}(x) U_2^T = U_2 \text{diag}(x_1, \dots, x_r) U_2^T,$$

where the parameter vector $x = [x_1, \dots, x_{cr}]^T$ is learned by solving (Eq. 3.21). Similarly, let the CSGE of A_2 be $A_2 \approx V_2 S_2 V_2^T$, then the prediction scores for **sl-csge** are given by

$$P_{\text{sl-csge}} = V_2 F_{\text{sl-csge}}(x) V_2^T = V_2 Q_2 \text{diag}(x_1, \dots, x_{cr}) Q_2^T V_2^T,$$

where Q_2 is obtained from the eigendecomposition $S_2 = Q_2 \Lambda_{S_2} Q_2^T$, and the parameter vector $x = [x_1, \dots, x_{cr}]^T$ is learned by solving the least squares problem in (Eq. 3.21).

3.3.3 Evaluation

In this section we present experimental results that evaluate accuracy, scalability, and flexibility of CSGE in the context of proximity approximation,

Network	Date	# of nodes	# of links	# of added links (in %)
Flickr	4/14/2007	1,990,149	41,302,536	–
	4/25/2007	1,990,149	42,056,754	754,218 (1.8%)
	5/6/2007	1,990,149	42,879,714	822,960 (1.9%)
LiveJournal	2/16/2009	1,770,961	83,663,478	–
	3/4/2009	1,770,961	84,413,542	750,064 (0.8%)
	4/03/2009	1,770,961	85,713,766	1,300,224 (1.5%)
MySpace	12/11/2008	2,137,264	90,333,122	–
	1/11/2009	2,137,264	90,979,264	646,142 (0.7%)
	2/14/2009	2,137,264	91,648,716	669,452 (0.7%)

Table 3.6: Summary of the online social network datasets.

missing link inference, and link prediction.

3.3.3.1 Dataset description

In our experiments we used three real-world large online social networks with millions of nodes: Flickr [36], LiveJournal [61] and MySpace [70]. LiveJournal and MySpace datasets are obtained from [88], and the Flickr dataset is collected by [68]. Table 3.6 summarizes some of the characteristics of three snapshots for each network. In Figure 3.7 we plot the shortest hop distance of user pairs in A_1 who become connected in A_2 (*i.e.*, positive set \mathcal{P}_1). We see that more than 70% of links in \mathcal{P}_1 are between user pairs who are two-hops away, whereas the fraction of new links between users who are 4 or more hops away is very small. All users in the datasets are connected to at least one other user and for simplicity, we do not consider rare occasions of link deletions. We also make Flickr and LiveJournal networks undirected to simplify the evaluation.

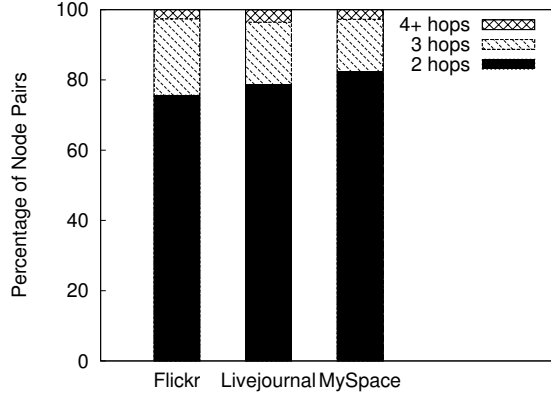


Figure 3.7: Shortest path distance of positive sets \mathcal{P}_1 .

Flickr [36] is a photo-sharing website, where users can connect to each other by indicating a relationship. This dataset was gathered by a breadth-first search on the graph starting from a few seed nodes. To allow most nodes to be discovered, we use the first few months as a bootstrap period, and create snapshots when most nodes have been discovered and link growth has stabilized. Because of this crawling methodology, we observe that even though the snapshot dates are just ten days apart, there is 2% growth in the number of links.

LiveJournal [61] is a blogging site, where members can become “fans” of other members. We obtained the LiveJournal dataset from [88]. The dataset was collected by listening to the RSS server, which sends out recent update information. The statistics suggest that LiveJournal users are more active in forming links than users in other networks.

MySpace [70] is a social networking site for people to interact with their

	c	$\mu = \sum_i m_i / c$	Links in A_{ii}	links in $A_{ij}, i \neq j$
Flickr	18	110,563	71.8%	28.2%
LiveJournal	17	106,241	72.5%	27.5%
MySpace	17	125,721	51.9%	48.1%

Table 3.7: Clustering results with Gracclus. c is the number of clusters and μ is the average cluster size.

acquaintances by posting on each other’s personal pages. We obtained this dataset from [88]. The graph is symmetric since both users need to agree to having a social link. The dataset was created by collecting information of the first 10 million user IDs. MySpace assigns user IDs chronologically. Since the first few million ids we crawled are also the oldest MySpace users, they have already formed most of the links and are relatively dormant in forming new ones (*i.e.*, the fraction of new links is smaller than those of other networks).

3.3.3.2 Scalability

In this section, we evaluate and compare various aspects of the computational efficiency of the CSGE, SGE and the *proximity embedding* introduced in [88]. All benchmarking was done using an Intel Xeon™ E5440 machine with 32GB memory, running Ubuntu Linux Kernel v2.6.

Graph clustering: In our experiments we use both GRACCLUS[27] and METIS [2] to partition the social networks. The two software packages produce different clusterings as they minimize different objective functions. For example, METIS attempts to produce clusters with equal size, regardless of the inherent clustering structure of the network, whereas GRACCLUS produces

Dataset	Prox. measure		SGE		CSGE						
	kz	rpr	A	T	clustering	SGE of all A_{ii}	S	Total	SGE of all T_{ii}	\bar{S}	Total
Flickr	216	276	19.9	2,821	1.4	12.0	2.3	15.7	327	3.1	331.5
LiveJournal	324	456	23.8	420	1.4	12.8	2.4	16.6	33.0	2.5	36.9
MySpace	330	588	22.8	546	2.6	14.4	2.9	19.9	48.1	3.5	54.2

Table 3.8: Comparison of preparation times ($c = 20$ and $r = r_i = 100$). All timings are in minutes.

Dataset	Direct Method		Proximity Embed.	SGE		CSGE		
	cn	kz	kz-prox.embed.	cn-sge	kz-sge	cn-csge	kz-csge	sl-csge
Flickr	15.9ms	8040ms	0.051ms	0.042ms	0.045ms	2.72ms	2.76ms	5.59ms
LiveJournal	23.5ms	14790ms	0.045ms	0.038ms	0.036ms	5.45ms	2.03ms	4.57ms
MySpace	24.5ms	16655ms	0.076ms	0.040ms	0.036ms	2.60ms	2.65ms	4.73ms

Table 3.9: Comparison of query times ($r = r_i = 100$, 0.6 million samples).

more balanced clusters. Both software produce good quality partitioning in short time. All experiments are conducted on the largest connected component of each graph as each connected component can be decoupled and dealt with independently. Only a small fraction of users and links are discarded by only working with the largest connected component. For example, the largest connected component of the Flickr network contains 93.9% of the users and 99.5% of all links. Table 3.7 gives an example of clustering results on each data set. In this particular case the clustering was performed using GRACLUS in a recursive way until all cluster sizes were smaller than 1/10 of original network size. We observe that more than 70% of the links in Flickr and LiveJournal are within the same clusters, while only 51.9% of the links in MySpace are within the same clusters.

Timing benchmarks: There are three steps involved in the CSGE: clustering, SGE of all clusters, and computation of the embedded adjacency matrix S . We present timings for each one of these steps and compare them with

timings for the SGE of the entire network. The CSGE and SGE are used to approximate both the adjacency matrix A and the normalized adjacency matrix $T = D^{-1/2}AD^{-1/2}$. We also present timings for the proximity embedding, which does not directly approximate A or T , but rather generates models to compute approximations of P_{kz} or P_{rpr} using relatively small number (1,600) of “landmark” nodes [88].

Timing benchmarks for the different methods for all three data sets are presented in Table 3.8. Clustering in these experiments is based on METIS using $c = 20$. Performance in terms of the number of links within the different clusters is similar to the results from Table 3.7. The rank in the SGE and each cluster in CSGE approximations was $r = r_i = 100$. We see that CSGE consistently outperforms the other two methods. In the approximation of A , CSGE is up to 30% faster than state of the art spectral embedding algorithms. In the approximation of the normalized adjacency matrix T , the timing difference between CSGE and SGE becomes even higher, resulting in over an order of magnitude difference. The convergence of iterative spectral methods is influenced by the size of the “gap” in the eigenvalues [7]. It is also well known that the eigenvalues of T are between 1 and -1 with very small gaps between them. Therefore, it is expected that the computation times for T become longer. The Flickr data set seems to be the extreme with respect to this. All 100 dominant eigenvalues of T and all 100 dominant eigenvalues of each cluster T_{ii} are very close to one.

In Table 3.9 we present the average time to approximate the different

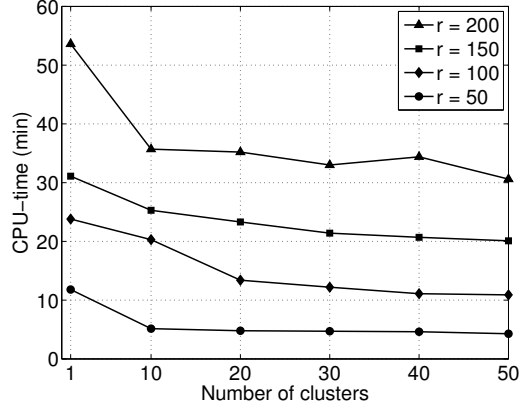


Figure 3.8: Comparison of eigen decomposition times.

proximity measures. The computations are performed on a sample set with 600,000 node pairs. Compared to the common neighbor and Katz measure, all the three embeddings are orders of magnitude faster. Recall that computing exact proximity measures or proximity approximations on all node pairs is not practical with large graphs, as the computed matrices will become dense. We note that once the different models are calculated the query times for all proximity measure approximations (based on both SGE and CSGE) are small and comparable to timings of the common neighbor method.

In Figure 3.8, we present aggregate times for the spectral embedding step of the CSGE method with varying number of clusters and varying size of the embedding dimensions. The timing results for $c = 1$ are simply SGE on the entire adjacency matrix A (since it is connected). Clustering time is not included. We observe from the graph that for any given r the aggregate time of cluster-wise approximations is significantly less than the time SGE. It

	A_{FL}	B	C
m	2,000,000	6,000,000	12,000,000
Numbers of links	40,000,000	239,000,000	598,000,000
c	18	52	104
Timing	22.6 min	116.3 min	391.2 min
Memory usage	74 MB	546 MB	1,197 MB

Table 3.10: Computational time and memory usage for the CSGE. Network size m and number of links are approximate.

is also evident from the curves that the general trend of computational time decreases with increasing number of clusters.

Scalability of graph embedding: In Section 3.3.1.2, we discussed the memory usage of the CSGE. To show the scalability, both in computational time and memory efficiency, we present some measurements on even bigger (artificial) networks. Let A_{FL} , A_{LJ} , and A_{MS} be the adjacency matrices for Flickr, LiveJournal and MySpace, respectively. Then we form

$$B = \text{diag}(A_{\text{FL}}, A_{\text{LJ}}, A_{\text{MS}}) + A_{\text{Off}}$$

where A_{Off} contains a small fraction of links in the off-diagonal part so that B becomes connected. We also formed C in a similar way as follows:

$$C = \text{diag}(B, B) + B_{\text{Off}}.$$

One may consider that the constructed datasets correspond to some real-world social networks. Approximate sizes, number of links, and number of clusters are presented in the upper half of Table 3.10. The lower part of Table 3.10

shows the memory required to store $V \text{diag}(V_1, \dots, V_c)$ and the embedded adjacency matrix S . Corresponding computational times are also given. The clustering for this experiments is given by the clustering of the individual networks. We fix $r_i = 100$ in all cases and examine the behavior by increasing the number of clusters c rather than the size of individual clusters. The memory and time required for creating the embedded adjacency matrix S grows quadratically with the number of clusters c , but also with the rank r_i . Despite the fact that the size of S is inherently quadratic to c , the CSGE is able to handle the largest network with 12,000,000 users and 598 million links - with only 1.2 GB of memory. On the other hand, the regular SGE is unable to load U for the B network, confirming that S is indeed not a limiting factor of memory usage in CSGE, and that it has higher spatial scalability than SGE.

3.3.3.3 Proximity measure estimation

Next we compare the accuracy of the Katz measure, rooted PageRank, and escape probability when they are combined with the SGE or CSGE. Due to spacial constraints, we present results only for the LiveJournal network.

Experiment setup. As the benchmark results in Table 3.9 suggest, it is impractical to directly compute proximity measures for all user pairs, *i.e.*, we can not afford to compute the dense matrices of $P_{\text{kz-sge}}$ or $P_{\text{kz-csge}}$ for all user pairs. Therefore, we evaluate the different methods using a sample set \mathcal{S} , which consists of 100 columns chosen randomly from the proximity matrix. This gives roughly 180,000,000 sample points. For each pair $(i, j) \in \mathcal{S}$ we let p_{ij} denote

the “true” value of a proximity measure and \hat{p}_{ij} to be an “estimate”. For example we may have $p_{ij} = P_{\text{kz}}[i, j]$ while $\hat{p}_{ij} = P_{\text{kz-sge}}[i, j]$. The true proximity measures were computed for comparison purpose using the methodology outlined in [88]. In all Katz measure computations, we use $\beta = 0.0005$ and $k_{\text{max}} = 6$. For computing the rooted PageRank and the escape probability, we use $\alpha = 0.15$ and $k_{\text{max}} = 20$. Recall that the different proximity measures are described in Table 3.5. The embedding subspace dimensions in the SGE and each cluster in CSGE are set to $r = 100$. Clustering of the datasets is done on A while the graph embedding is based on A for the Katz measure and $T = D^{-1/2}AD^{-1/2}$ for the rooted PageRank and the escape probability measures.

Metrics. As for the accuracy measure, we use *normalized absolute error* (NAE), defined by $e_{ij} = |p_{ij} - \hat{p}_{ij}|/\mu$, where $\mu = \sum_{(i,j) \in \mathcal{S}} p_{ij}/|\mathcal{S}|$. For each method and a set of computed NAEs, we plot the *cumulative distribution function* (CDF).

Accuracy evaluation

Figure 3.9 (a) plots the CDF of the normalized absolute errors in approximating the Katz measure with $P_{\text{kz-sge}}$ and $P_{\text{kz-csge}}$. We make two observations: (1) for most samples the normalized absolute error is small: in 85% of the node pairs, the error is 0.2 or less; and (2) the error for the CSGE is lower than SGE while the gap is as small as 2%. A likely reason for SGE and CSGE yielding similar performance is that P_{kz} has low intrinsic dimensionality and

both SGE and CSGE have reached to a point where $r = 100$ is enough for low-rank approximation of Katz metric.

In Figure 3.9 (b) we present the CDF of normalized absolute errors in approximating the rooted PageRank measure with $P_{\text{rpr-sge}}$ and $P_{\text{rpr-csge}}$. We observe that clustering gives a considerable improvement of the accuracy of the rooted PageRank measure. For over 95% of the samples, the normalized absolute error is less than 0.01. On the other hand, SGE exhibits a relatively higher error of 0.6 for 95% of the samples. It can be verified that the normalized adjacency matrix T , for which we compute SGE and CSGE, has a much higher intrinsic dimensionality than A . The improved accuracy in $P_{\text{rpr-csge}}$ may be explained by the fact that 100 dimensional embeddings on each cluster (CSGE) captures a much larger fraction of variance compared to an $r = 100$ embedding on the entire matrix (SGE). Figure 3.9 (c) shows the CDFs of normalized absolute error for escape probability measure $P_{\text{ep-sge}}$ and $P_{\text{ep-csge}}$. Again, using clustering significantly improves the accuracy. This improvement is to be expected as escape probability is based on the rooted PageRank measure. While CSGE provides normalized absolute error less than 0.1 for more than 95% of the samples, SGE exhibits much higher error of over 20 for the same 95% of the samples.

To summarize, our proximity estimation evaluation shows that CSGE is not only effective in approximating proximity measures on A but also performs well on matrices with high intrinsic dimension such as the normalized adjacency matrix T . Compared with SGE, CSGE can accurately approximate

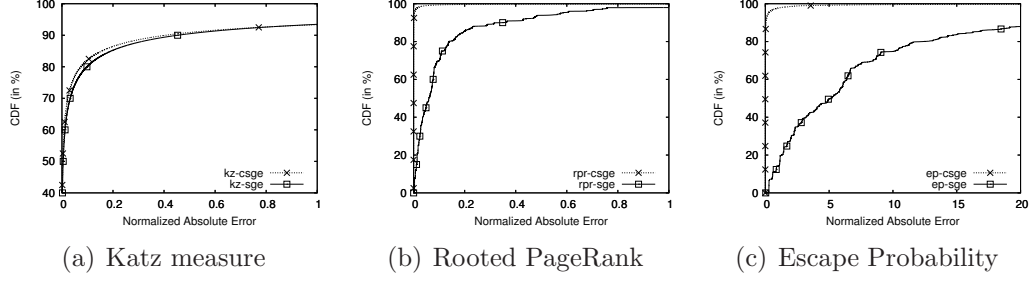


Figure 3.9: Accuracy of proximity estimation for LiveJournal dataset.

the rooted PageRank and the escape probability measures even though these metrics are known to be difficult to approximate [88].

3.3.3.4 Missing link inference

An important problem in social network applications is missing link inference. We evaluate the accuracy of inferring different proximity measures with and without clustering.

Metrics. The accuracy of missing link inference is measured by computing the false negative rate (FNR), and the false positive rate (FPR), defined by

$$\begin{aligned} \text{FNR} &= \frac{\text{\#of missed missing links}}{\text{\#of actual missing links}}, \\ \text{FPR} &= \frac{\text{\#of incorrectly predicted missing links}}{\text{\#of non-friend pairs}}, \end{aligned}$$

or all user pairs in a sample set. Note that the denominator of the FPR is the number of user pairs who are not friends. This number is usually very large, *e.g.* in MySpace we have $2 \times 10^6 \times 2 \times 10^6 - 90 \times 10^6 \approx 10^{12}$. To focus on the more interesting region (with low FPR) in the plots, we present the FPR along the x -axis in log-scale, and FNR along the y -axis in regular scale.

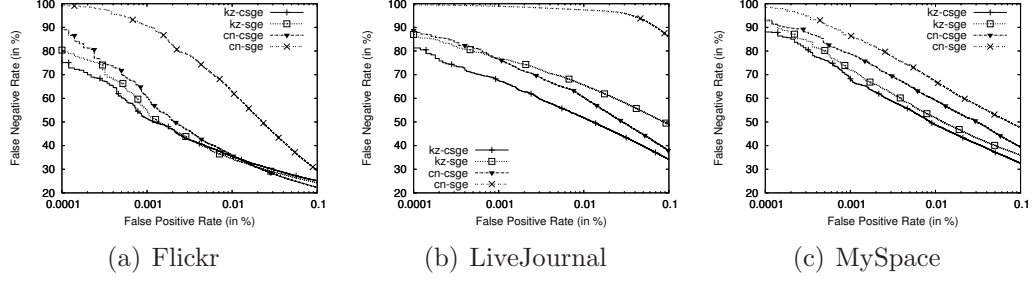


Figure 3.10: Missing link inference accuracy for different proximity measures.

Experimental setup. From the first snapshots of all three datasets, we randomly mask half of the links as missing \mathcal{M} . The adjacency matrix with the remaining links is considered as an incomplete adjacency matrix with observed links A_0 . The trade-off curve between FPR and FNR is plotted for all user pairs given the non-zero of $A_{\mathcal{M}}$.

Accuracy evaluation In Figure 3.10, we present the performance of $P_{\text{cn-sge}}$, $P_{\text{cn-csge}}$, $P_{\text{kz-csge}}$, and $P_{\text{kz-sge}}$. We observe that using CSGE in both proximity measures consistently outperforms SGE in all three datasets. Comparing across different measures, we observe that Katz measure with CSGE generally performs the best. For example, using the LiveJournal data set, for a given FPR, $P_{\text{kz-csge}}$ yields 10% or less FNR than $P_{\text{kz-sge}}$.

3.3.3.5 Link prediction evaluation

In this section, we compare the performance of supervised clustered and non-clustered spectral link prediction method against unsupervised proximity measures. Through the performance evaluation, we verify if the training of

control parameters done in supervised measure is indeed helpful in improving accuracy within all datasets.

Training and testing steps. Table 3.6 shows the brief information related to the three snapshots A_1 , A_2 , and A_3 of each dataset. To learn the model parameters in the the two supervised models, we use a graph embedding of A_1 and minimize the corresponding objective function by explicitly targeting newly formed links in A_2 . In the next step, we use an embedding of A_2 with the learned model parameters to predict new links in A_3 .

Since our datasets are very large, we randomly select a fraction of the positive samples and negative a fraction of the samples. Specifically, for training, we select 100,000 user pairs from the positive links \mathcal{P}_1 , and 500,000 user pairs from the negative links \mathcal{N}_1 . For testing, we pick a different sample set of the same size as before but now from \mathcal{P}_2 and \mathcal{N}_2 .

In a second experiment, we learn the model parameters based on a sample set of users pairs that are connected by two hops. This practical scenario focuses on link prediction for user pairs who are already close in the network (but not friends yet). These are user pairs that will form a triangle with a common friend, if they become friends.

Metrics. The accuracy of link prediction is quantified using the FPR and FNR introduced in Section 3.3.3.4. In the context of link prediction, the “true” and “estimated” links refer to the links in \mathcal{P}_2 . As in the missing link inference, the raw count of non-friend pairs in FPR is extremely large. Since

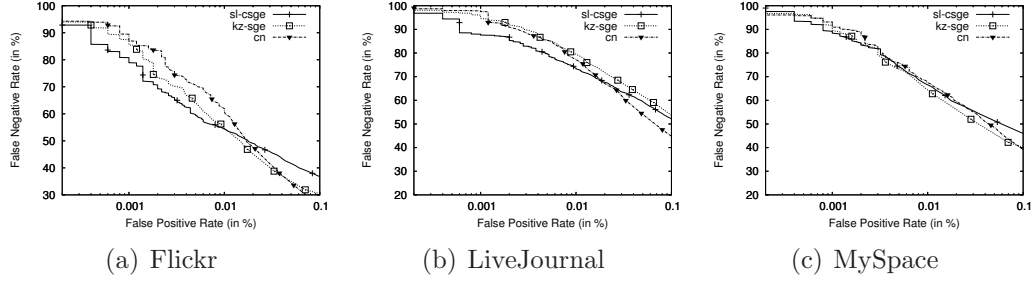


Figure 3.11: Link prediction accuracy of different measures.

we are more interested in picking up small number of correct friendships (as opposed to finding as many new friendship links as possible), we present the performance measures with small FPR values area in the trade off curves.

Link predictors. we have conducted extensive set of experiments, with many different methods. In the interest of brevity, and not to clutter the figures too much, we will present link prediction results based on three link models: P_{cn} , P_{kz-sge} and the supervised $P_{sl-csge}$.

Clustered Spectral Graph Embedding. Figure 3.11 presents the link prediction accuracy of P_{cn} , P_{kz-sge} and $P_{sl-csge}$. We see that, the spectral learning with CSGE performs the best in all datasets, followed by the Katz measure with SGE. For example, at an FPR of 0.001, spectral learning with CSGE reduces FNR by more than 10% in Flickr and LiveJournal. In MySpace, the performance of clustered spectral learning is still better than the other two measures albeit the gain is less 3% less FNR).

Two hop user pairs. In figure 3.12, we evaluate link predictors for a different setup: user pairs who are only two hops away. Our spectral learning model

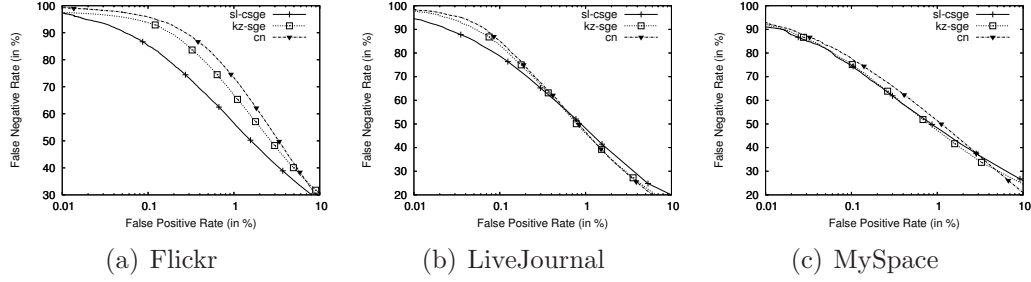


Figure 3.12: Link prediction accuracy with 2-hop scenario.

again outperforms other measures by up-to 20% in the Flickr dataset, and 10% in the LiveJournal dataset. For MySpace, the performance of Katz with SGE and spectral learning with CSGE is better than common neighbor by 4%.

Note that there is no significant difference among predictors in MySpace. We speculate that it is an artifact of the data collection technique that takes the first 10 million users. Since the first users have been in the network the longest time, they may already have a large number of friends, and thus they are less active in creating new relationships. An indication of the claim is the the small increase of the links between two snapshots compared to the number of existing links (shown in Table 3.6) and other networks. Another reason may be the large number of inter-cluster links in MySpace, as shown in table 3.7. Taking into account that 48% of the links are outside the clusters, the benefit of clustering approach may have diminished, as almost half the links in the network are not used when computing the cluster-wise embeddings. The overall CSGE result for this particular experiment is similar to SGE.

Summary. This section shows that our spectral learning algorithm is flexible and consistently out-performs the existing schemes across different datasets and proximity measures.

3.4 Summary of Social Network Analysis

In this section, we illustrate two novel approaches to analyzing node proximity in social networks. In the first work, scalable proximity inversion, we develop several novel techniques to approximate a sub family of proximity measures in massive, highly dynamic online social networks.

In the second scheme, clustered spectral graph embedding (CSGE), we develop a novel dimensionality reduction technique that can embed a massive original graph into a much smaller graph. The embedded graph captures the essential clustering and spectral structure of the original graph and provides flexibility that a wider range of proximity measures to be approximated in scale.

Chapter 4

Proactive Service Quality Assessment

Recent advances in residential broadband access technologies have led to a wave of commercial application network deployments. One such cutting edge application network is Internet Protocol TV (IPTV) system. IPTV technologies are transforming the global television industry, enabling new operators to provide TV services whilst also enabling a wealth of innovative new IP-based services integrated with more traditional TV. However, there is a pressing need to ensure that the IPTV services being deployed deliver a Quality of Experience (QoE) that is equal to or better than traditional TV services.

The traditional approach to assessing quality of experience has been through subjective evaluation in controlled laboratory environments. Subjective evaluation is expensive, error-prone and unreliable. A complementary approach is through user feedback about the service quality collected from deployed services. User feedback in the form of complaints is collected by the call centers and it provides a direct measure of the service performance problems experienced by the users. However, user feedback is incomplete (not all users complain) and delayed (damage is already done).

What operators lack today is a proactive approach to obtaining comprehensive views of user.s quality of experience. Such views are critical to proactively detecting service issues that matter to customers so that operators can rapidly respond to them to ensure a high-quality customer experience. Without such visibility, operators risk being unaware of service degradations - instead learning about issues only as customers reach frustration points and complain. Thus, proactive assessment of quality of experience is crucial to providing operators with insights into the ultimate metric - what customers are experiencing - so that they can effectively manage their service offerings and detect and ideally respond to issues *before customers complain*. Proactive assessment of quality of experience can also help operators in effective dimensioning of the customer care workforce in anticipation of the large volume of user complaints should unavoidable, customer-impacting conditions arise.

Although there is extensive monitoring of network elements in place today and operators rapidly react to issues which are reported by the network elements - there is no technology which can directly measure *customer perception* of TV service quality. Video monitoring technologies exist, but it is still non-trivial to relate such measurements to customer perception. Deploying video monitoring to millions of customers is also often prohibitively expensive, and service providers are also typically constrained by the technology available within the Set Top Boxes.

In the absence of direct measurements, we instead focus on using network measurements to infer customer service experience. However, such an

approach is still challenging, due to (i) incomplete knowledge about the relationship between user-perceived issues and network performance metrics, and (ii) user feedback about quality of experience is biased towards the negative (i.e., customer complaints) and is often delayed, noisy and limited which makes the inference even more challenging.

We propose a new framework, which we refer to as *Q-score* [89], for proactive assessment of user’s quality of experience. Q-score constructs a single quality of experience score using performance metrics collected from the network. Q-score consists of two key components: (i) offline learning of the association between the service quality of experience and the network performance metrics collected from the servers, routers and in-home equipment, and (ii) online computation of the score for individual users or groups of users. Q-score captures the quality of experience by users in a timely fashion and can provide operators with rapid notification of service issues, often giving them a lead time of several hours before the user complains to the call center.

Q-score Design Challenges. Due to the interwoven server and network system, as well as the sophisticated hardware and software composition of home network devices, assessing service quality of experience is a sophisticated task. The proposed Q-score approach uses customer complaints (e.g., tickets) to provide feedback regarding issues that customers are concerned about. Designing Q-score required us to address the following key challenges:

1. **Difficulty in associating QoE with network performance.** Be-

cause of the inherent difference between network-level performance indicators and user-perceived quality of service, associating the two does not occur naturally. Even for domain experts, since there is no objective video quality metric, it is not trivial to identify key performance indicators that are closely related to quality of experience (QoE). Even if the metric were available, it would require more finely grained monitoring of network indicators, which in turn might introduce scalability issues.

2. **Lack of timely, high-quality user feedback.** User feedback is inherently noisy, incomplete and delayed. Some users issue a complaint immediately after they observe a service quality degradation; others may take hours to complain. Similarly, different users have different tolerance levels to service quality issues - one user may complain incessantly regarding issues that another user may barely notice. Furthermore, the amount of delay in reporting service quality issues is variable. Depending on situations such as the individual viewer's living schedule, the severity of the issue, there are large variances between the beginning of service quality issues and reporting times. This all makes it inherently challenging to use such feedback to associate service quality of experience with network performance.

3. **Large volume of diverse performance measurements.** From a network perspective, service providers typically collect fine-grained measurements from the routers and servers (*e.g.*, real-time syslogs, and regular polls of SNMP performance counters such as CPU, memory, packet

counts, and losses). Some performance measurements inside the home may be fine-grained (*e.g.*, residential gateway events), whereas others may be coarse-grained (*e.g.*, hourly or daily summaries of Set-Top-Box events). Set-Top-Box (STB) data collection is intentionally not fine-grained to minimize the potential of service disruption due to measurements and due to the massive scale of the measurement infrastructure that would be required. The diversity in the granularity of performance measurements poses interesting technical challenges in inferring the quality of experience.

Our Contributions.

1. We design and implement a prototype Q-score system for proactively assessing quality of experience for IPTV users. Q-score uses a multi-scale spatio-temporal statistical mining technique for computing a single score capturing the quality of experience. By performing spatio-temporal aggregation and multi-scale association of the user feedback with network performance metrics, it identifies the right set of metrics useful for accurately quantifying the quality of experience.
2. We evaluate Q-score using data collected from a large commercial IPTV service provider and show that Q-score is able to predict 60% of customer service complaints with 0.1% of false positives.
3. We create three applications above Q-score. (i) *Identifying important KPIs* that are statistically associated with the quality of experience ,

(ii) *Predicting bad quality of experience* to users and generating alerts to the Operations team, and (iii) *Effective dimensioning of the customer care workforce* to dynamically allocate repair personnel to service regions as they experience issues for conducting root-cause diagnosis and rapid repair.

4.1 Background

In this section, we give an overview of IPTV service architecture followed by a detailed description of the data sets used in this chapter.

4.1.1 IPTV Service Architecture

Figure 4.1 provides a schematic overview of an IPTV system. The service network exhibits a hierarchical structure where video contents are delivered from the servers in core provider network to millions of Set-Top Boxes (STBs) within home networks via IP multicast. Specifically, either Super Head-end Office (SHO) which serves as the primary source of national contents or Video Head-end Offices (VHOs) which governs local contents at each metropolitan areas encodes, packetizes and sends the contents towards end users. Depending on the service provider, the video contents go through several routers and switches in Intermediate Offices (IOs), Central Offices (COs), Digital Subscriber Line Access Multiplexer (DSLAM), and Residential Gateway (RG) before reaching STBs where the packetized contents gets decoded and displayed on the TVs. All of the network entities comprising IPTV ser-

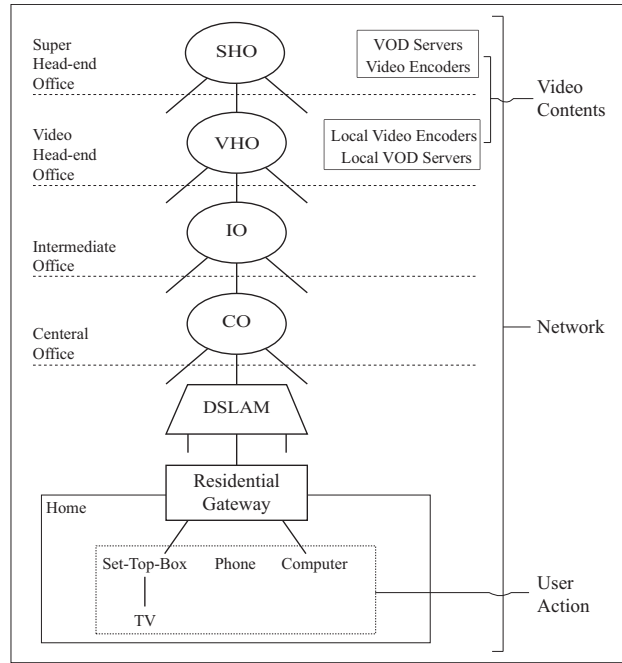


Figure 4.1: IPTV service architecture

vice logs key performance indicators (KPIs) such as delivery status of data and health diagnostics.

4.1.2 Data Sets

In this work, we use data collected from a large commercial IPTV service provider in the United States, which has customers spread throughout four different time-zones. Our data set consists of (i) network performance indicators, (ii) user behaviors and activities, and (ii) user feedback in form of customer complaints. We normalize timestamps in all data sets to GMT to accurately and effectively associate the user feedback with performance metrics and user behaviors.

Data set Type	Spatial Level	Description
Network Performance Indicators	STB	STB audio quality indicators
		STB video quality indicators
		STB syslog
		STB resets
		STB crashes
	RG	RG Reboots
	IO & CO	SNMP MIBs of routers & switches
	SHO & VHO	SNMP MIBs of routers & switches
User Interaction Indicators	User	STB power on/off log
		STB Channel change log
		STB Stream control log
User Feedback	User	Customer trouble tickets

Table 4.1: Summary of Data sets

Network Performance Indicators. Network performance indicators are categorized into two parts: (i) provider network performance indicators, which are collected from routers and switches in SHO, VHO, IO, CO of the IPTV service provider as shown in Figure 4.1 and (ii) home network performance indicators, which are collected from components inside user’s homes (i.e., RG and STB). For the provider network performance data, we collected SNMP MIBs from every router and switch in SHO, VHO, IO, and CO. The SNMP MIBs report five minute average performance statistics of CPU utilization and fifteen minute average summaries for packet count, packet delivery errors and discards.

From the home network side, we first collected data relevant to STB and RG. STB records audio and video streaming-related information including video throughput, receiver transport stream errors, codec errors, DRM errors,

and viewing duration of TV channels. The video streaming-related information is reset when the TV tuner clears its buffer by switching channels. While each STB logs all the TV viewing information at all times, polling servers only take a subset of STBs' statistics at each polling interval (due to high volume of audio and video log and traffic overhead during data delivery). As a result, only a sampled set of STBs are used in our study. Secondly, we collected STB syslog information that contains a wide variety of hardware and software information such as hard disk usage and memory usage, data delivery status such as packet error rate and buffer usage. The diagnostic information are collected in the same way as the STB audio and video log, *i.e.*, polled by collection server in round robin fashion. Thirdly, we collected crash and reset events log from each STB. The crash events log refers to unexpected rebooting of STBs due to software malfunctions and the reset refers to intentional and scheduled reboots commanded by network operators due to, for instance, software updates. The crash and reset log are periodically collected from all STBs with millisecond scale time stamps. Last performance indicator taken from home network is the reboot log of RGs that are commanded by operators remotely. RG reboot log is collected in the same way as STB reboot log.

User Activity Data. Because IPTV networks are highly user-interactive systems, certain user activity patterns or habits can create overload conditions on the STB and cause a service issue (*e.g.*, a user changing channels too frequently may cause its upstream device such as a DSLAM to be overwhelmed leading to inability in handling all the remaining STBs that it serves).

Hence, user activities are another important factor to be considered. Similar to conventional TV users, IPTV users use a vendor/provider-customized remote controller to control the STB. For this work, we collected logs from every STB that captures four types of user activities performed: (i) power on/off: this is the result of the user pressing the power button to turn on or off the STB; (ii) channel switch: this can be the result of one of the three actions: target switching by directly inputting the channel number, sequential scanning by pressing the Up/Down button, or pre-configured favorite channel list; (iii) video stream control: this includes actions such as fast forward, rewind, pause and play that are performed on either live TV streams, VoD, or DVR; and (iv) on-screen menu invocation: this log saves the user action of pulling up the STB menu displayed on TV screen that lets the users to access the features provided by the IPTV system.

User Feedback. For user feedback, we used complaints made to the customer care center of an IPTV service. Customer care cases are records of user interactions at call centers. A customer call can be related to service provisioning, billing and accounting, or service disruption. Since the focus of our work is on quality of experience (QoE), we specifically examined users' reports on service disruptions that later involved technical support as our user feedback. Each customer complaint related to service disruption includes the user ID, report date and time, brief description of the problem, and resolution of the issue.

4.2 Q-score Design

In this section, we introduce our proposed scheme, Q-score. The high level idea is to extract useful association between the noisy, incomplete, and indeterminately-delayed user feedback and the various network (including the servers, transport and in-home devices) performance indicators through an offline learning process, and then transform the knowledge into an online run-time system that estimates/predicts user-perceived service quality based on the available network KPIs. We start by giving an overview of the Q-score system architecture and then dive into details of each component.

4.2.1 Overview

Figure 4.2 presents the system architecture of Q-score. As shown in the figure, Q-score takes input from network (including servers, transport and in-home devices) performance indicators, which we refer to as *features*, the user control activities, and the user feedback in the form of customer call service records. The output is series of Q-scores, one for each user in service, quantifying the received service quality. At a high level, our system is composed of two components: (i) offline learning component and (ii) online continuous monitoring component. The overall dataflow in Q-score system begins with the offline relationship learning between user feedback on service quality and the measurements from network features and user activities. Ideally, if there had been any accurate and fine-grained user-level service quality measure, we would use it to train a model for network feature selection. However, as stated

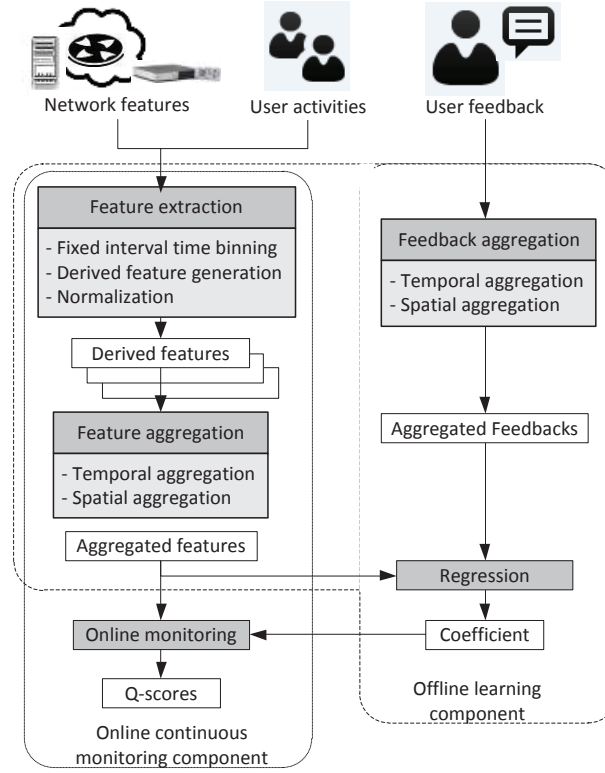


Figure 4.2: Overview of Q-score design

earlier, the best available channel for discovering user-level service quality issue is through the lossy, noisy and indeterminately-delayed calls to customer care centers. Consequently, we need to carefully design the appropriate temporal and spatial aggregations to remedy the inherent loss, noise and delay with user feedback. Furthermore, by applying statistical regression over large quantity of historical data between various network KPIs and the user feedback, we obtain a set of regression coefficients which quantitatively capture their relationship. These regression coefficients are fed into the online monitoring component.

With the regression coefficients, we can turn the up-to-date network KPI measurements into a single numerical score for each user or groups of them within a given spatial region. The numerical score, which we refer to as the *Q-score*, captures the likelihood of any on-going service quality problem. Tracking the Q-score over time enables many service management applications as will be described in Section 4.4.

4.2.2 Spatio-Temporal Feature Extraction

On each of the network performance indicators and user interaction indicators described in Section 4.1.2, we apply the following series of transformation to obtain a measurement matrix.

4.2.2.1 Transformation of Measurement Readings

Conversion to fixed-intervaled time bins.

Network measurement data collected from different sources and devices are bound to different time periods, posing challenge in correlating them. Some datasets, such as CPU level of routers in SNMP MIBs, contain periodically collected measurement data, and the value represents the average or total over the measurement interval. Some other datasets, such as user activities to STB and STB crashes logs, contain events that take place at a single point of time, hence are intermittent and have zero duration. Datasets such as STB audio and video quality indicators contain data polled either on demand or at irregular intervals and represent the cumulative counters over a variable time

interval (e.g., due to channel switches clearing the diagnostic counter entries).

To unify the data representation, we define a data point $d(m, l, s, e) = v$ as composed in a four dimensional specification: (i) metric $m \in M$ where M is a set of metrics such as CPU level of routers and count of video decoding errors at STBs. (ii) location $l \in L$ where L is a set of spatial location identifiers such as a set of users, DSLAMs, or COs. (iii) beginning time for the data binding interval $s \in T$, where T is the total time window, and (iv) ending time of the data binding interval $e \in T$. v is the measurement value that d contains. Note that for measurement data pertaining to a single time point, $s = e$.

The above representation is comprehensive in capturing various cases of periodical/intermittent or fixed/variable duration measurements. However, it requires a moderate amount of computation to determine the overlaps among the time intervals, which becomes prohibitively expensive for a large dataset as in our case. To reduce the complexity, we convert all $d(m, l, s, e)$ into a fixed-size time interval data representation $b(m, l, s, \delta)$ as follows:

$$b(m, l, s, \delta) = \{v \mid v = d(m, \bar{l}, \bar{s}, \bar{e}), \text{ where } l = \bar{l} \text{ and } [\bar{s}, \bar{e}] \text{ overlaps with } [s, s + \delta]\} \quad (4.1)$$

where δ is length of the feature time interval. Note that if there exist two or more ds with matching measurement time to $[s, s + \delta]$, there could also be multiple identical values for b – making b not well defined. We need to introduce the aggregation functions as below.

Conversion to derived features. To deal with multiple ds colliding into the

same b (either due to time bin or spatial aggregation), we define three types of aggregate data points, which we refer to as the *derived features*. They contain (i) the minimum, (ii) the maximum, and (iii) the average of all the values for b respectively. Formally,

$$f_m(m, l, s, \delta) = \min_{l \in \text{child}(\bar{l})} (\cup(b(m, \bar{l}, s, \delta))).$$

$$f_M(m, l, s, \delta) = \max_{l \in \text{child}(\bar{l})} (\cup(b(m, \bar{l}, s, \delta))). \quad (4.2)$$

$$f_a(m, l, s, \delta) = \text{avg}_{l \in \text{child}(\bar{l})} (\cup(b(m, \bar{l}, s, \delta))). \quad (4.3)$$

In this way we can limit the number of derived features to be three regardless of the number of actual readings in b . Unless specified otherwise, all features referred in the rest of the work are the derived features.

Feature normalization. Network features we consider typically take numerical values, potentially having different signs and across large range of scales. This makes it difficult to assess the significance of their associated coefficient under regression. To deal with the diverse data values, we further normalize features to be binary-valued by comparing to a threshold, which is determined depending on the metric and location.

Consider a vector of features of the same metric and location over different time and interval combinations:

$$\vec{f}_a(m, l) = \langle f_a(\bar{m}, \bar{l}, s, \delta) \text{ where } m = \bar{m}, l = \bar{l} \rangle \quad (4.4)$$

We need to identify a threshold value τ for \vec{f}_a . To do so, we bring in the user feedback in the form of user complaint logs. We consider the conditional

distribution function of the metric value of interest when (1) there is one or more entries of the user complaint log being associated with the location l and when (2) there is no such entry. Ideally, a threshold τ can separate the instances between case 1 and 2. When threshold τ is low, the chance of having instances in case 1 passing the threshold increases, and when threshold is high, the chance of having instances in case 2 failing the threshold increases. So, we set the threshold τ such that the two factors balance out. Using empirical CDFs of the case 1 (F_1) and case 2 (F_2), we can define τ such that

$$F_1(\tau) = 1 - F_2(\tau). \quad (4.5)$$

Once τ is determined, we can normalize of f_a as follows.

$$f_a(m, l, s, \delta) = \begin{cases} 1 & \text{if } f_a(m, l, s, \delta) \geq \tau \\ 0 & \text{otherwise.} \end{cases} \quad (4.6)$$

Features f_m and f_M can be normalized in the same way.

4.2.2.2 Construction of Measurement Matrix

In order to support multi-scale analysis that accounts for the indeterminate delay in user feedback, we construct the regression input matrix \mathbf{X}

over all measurement metrics, location, and time parameters as below.

$$\mathbf{X} = \begin{bmatrix} \begin{bmatrix} f_m(m_1, l_1, s_1, \delta) f_M f_a \\ f_m(m_1, l_1, s_2, \delta) f_M f_a \\ \vdots \\ f_m(m_1, l_1, s_t, \delta) f_M f_a \\ f_m(m_1, l_2, s_1, \delta) f_M f_a \\ f_m(m_1, l_2, s_2, \delta) f_M f_a \\ \vdots \\ f_m(m_1, l_2, s_t, \delta) f_M f_a \\ \vdots \end{bmatrix} & \begin{bmatrix} f_m(m_2, l_1, s_1, \delta) f_M f_a \\ f_m(m_2, l_1, s_2, \delta) f_M f_a \\ \vdots \\ f_m(m_2, l_1, s_t, \delta) f_M f_a \\ f_m(m_2, l_2, s_1, \delta) f_M f_a \\ f_m(m_2, l_2, s_2, \delta) f_M f_a \\ \vdots \\ f_m(m_2, l_2, s_t, \delta) f_M f_a \\ \vdots \end{bmatrix} & \dots \end{bmatrix} \quad (4.7)$$

The columns of \mathbf{X} represent different metric of derived features. Thus, each column has f with a unique m_i . The rows of \mathbf{X} represent all feature values during a specific time (s_i, δ) at a specific location l_j . Assuming there are n locations, t different time bins, and k different KPI metrics and feature aggregations, the number of rows in \mathbf{X} is $n \times t$ and the number of columns is k .

4.2.2.3 Multi-scale Temporal Level Aggregation

The time window parameter δ plays an important role in capturing the extend of cause-effect delays. Large δ would include cause-effect relationship with long delay. However, large δ would make it insensitive to dense measurements with short cause-effect delay, as the aggregation weakens the significance of correlation. Since different δ values have advantages over others, we adopt a multi-scale analysis approach by including multiple time window parameters into our consideration. Our matrix representation in Eq (4.7) is flexible enough to enable this – we append in columns the $\mathbf{X}(\delta_i)$ s with different time-intervals

(δ_i) .

$$\mathbf{X}_{\text{Temp.Comb.}} = [\mathbf{X}(\delta_1) \cdots \mathbf{X}(\delta_v)] \quad (4.8)$$

where v is the number of different values of the time window parameter.

4.2.2.4 Multi-scale Spatial Level Aggregation

Similarly to the temporal aggregation captured by the time window parameter, there can be multiple spatial aggregation levels with IPTV system architecture. Based on the hierarchical structure in Figure 4.1, we consider three different spatial aggregation levels in Q-score, namely user, DSLAM, and CO levels.

Single-scale Spatial Level Aggregation. We set the baseline spatial aggregation level to per user aggregation. This is because the service complaint logs are associated with a household, which we loosely refer to as a user. Matching the network features to the household/user level, one of the following processing is necessary: (i) for features at finer grained spatial level than user (such as STB related features since one household may have multiple STBs), we take the maximum among different feature values for the more specific locations as the representation for f_M , the minimum for f_m , and the average for f_a , at the user level; (ii) for features with coarser grained spatial level than user (such as DSLAM and CO), we replicate the coarser grained feature values for each associated user within the hierarchy. In this way, we preserve the number of samples to be $n \times t$ in each row of \mathbf{X}_{user} . The same spatial level aggregation is applied for DSLAM level and CO level to obtain $\mathbf{X}_{\text{DSLAM}}$ and \mathbf{X}_{CO}

respectively.

Multi-scale Spatial Level Aggregation. In parallel with the multi-scale analysis with respect to time window parameter, different spatial aggregation levels can be fed into regression altogether. The idea is that the most prominent feature would be at a suitable spatial aggregation level and would dominate the same features aggregated at other spatial levels. We append in column the feature matrices for different spatial levels to get the $\mathbf{X}_{\text{Spat.Comb.}}$:

$$\mathbf{X}_{\text{Spat.Comb.}} = [\mathbf{X}_{\text{userID}} \ \mathbf{X}_{\text{DSLAM}} \ \mathbf{X}_{\text{CO}}]. \quad (4.9)$$

4.2.3 Feedback Aggregation

As outlined in Section 4.1.2, we use the user complaint logs through customer service calls as the user feedback regarding service quality. This feedback is inherently unreliable. It is incomplete as not all service quality problems (e.g., video glitches) would be noticed by user – when the user is not actively watching the TV for example. Different users vary in their tolerance level of video problem and in their readiness to call customer service to report/complain about the problem, making this feedback very noisy. Furthermore, users may not pick up the phone at the first sight of the service quality degradation. There is an indeterminate delay ranging from minutes to hours to even days between the service problem and the user complaint log entry. All of these require some denoise processing for such user feedback to be useful even in statistical sense.

We adopt the same principle applied in the spatio-temporal aggregation

with respect to network features. Let c be the predicate of the presence of a matching entry in the feedback log (B):

$$c(l, u, \gamma) = \begin{cases} 1 & \text{if } \exists b \in B \text{ during } [u, u + \gamma]; \\ 0 & \text{otherwise.} \end{cases} \quad (4.10)$$

where u is the beginning time for a feedback binding interval and γ is the length of feedback time interval. Once $c(l, u, \gamma)$ is defined, we can use the same spatio-temporal aggregation method for the network features on c .

A network event or user activity is always a cause of user feedback but cannot be an effect. Thus we set $u = s + \delta$ so that when we correlate c_i to b_i , we take account of the causal sequence between network (or user activity) events and user feedback. Let y be a vector of feedback for different users over time

$$\mathbf{y} = [c(l_1, u_1), \dots, c(l_1, u_t), c(l_2, u_1), \dots, c(l_2, u_t), \dots]^T.$$

The length of the vector \mathbf{y} is determined by the number of locations n and the number of time bins t , making it to be $n \times t$ which is the same as the row count of \mathbf{X} .

4.2.4 Regression

Given the measurements of network indicators \mathbf{X} and user feedback \mathbf{y} , we now aim to find a coefficient vector β that provides a compressed representation of the relationship between \mathbf{X} and \mathbf{y} .

Such optimization can be performed using *regression*. A baseline regression model of linear regression [14], however, cannot provide the opti-

mal solution as our system of equation $\mathbf{X}\beta = \mathbf{y}$ is over-constrained (*i.e.*, the equation has far smaller number of unknowns than the number of equations ($k \ll (m * n)$)). To prevent β from over-fitting due to high variance, we apply Ridge regression [46] that imposes penalty λ on complexity of model by minimizing a penalized residual sum of squares \mathcal{RSS} as follows

$$\min_{\beta} \mathcal{RSS}(\mathcal{D}, \beta) \text{ s.t. } \sum_{i=1}^n \beta^2 \leq s. \quad (4.11)$$

where \mathcal{D} is the set of observed data points $\mathcal{D} = x_n, y_n$.

We can state this optimization problem in Ridge regression as

$$\hat{\beta} = \arg \min_{\beta} \sum_i (y_i - \beta_0 - \sum_{j=1}^p x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^p \beta_j^2. \quad (4.12)$$

The Ridge coefficient $\hat{\beta}$ becomes

$$\hat{\beta} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}. \quad (4.13)$$

where \mathbf{I} is the identity matrix.

There are other regression methods we have explored including l_1 -norm minimization and logistic regression. However, as our system of equation has tens of thousands of equations and thousands of unknowns, l_1 -norm minimization and logistic regression either took excessive amount of time in computation or failed to converge to an answer. The complexity and scale of our system make these techniques infeasible.

Finding significant KPI weights From the β coefficients, we can identify key performance indicators (KPIs) that are more closely related to user feedback. This involves sorting the regression coefficients by their absolute value

and identifying the top N KPIs associated with them. Furthermore, by analyzing the commonality and difference of the same metric across different temporal and spatial aggregation configuration, we can gain insight on how each of these KPIs impact the users' quality of experience specific to the most significant spatio-temporal aggregation. The analytical results on the most significant KPIs in IPTV are presented in Section 4.4.1.

4.2.5 Compute Q-score in Runtime

Once the offline learning of β completes, we can compute from the available key performance indicators the Q-scores either for individual users or groups of users aggregated spatially depending on the feedback aggregation scheme used.

Detecting significant Q-score changes With β from the offline learning, we apply it to the current network measurement data \mathbf{X} and obtain Q-score that estimates per user level service quality. Running continuously as network KPI data streaming into Q-score, we track the series of Q-scores over time.

Since Q-scores are real-valued numbers, we need to identify the thresholds for alarming on the Q-scores to the operations. The alarms can be proactively used to predict customer complaints. We apply simple threshold-based change detection on the time-series of Q-scores to generate the alarms.

4.3 Evaluation

In this section, we present the performance evaluation results of Q-score and show that the regression results are accurate and robust, and the multi-scale aggregation of spatio-temporal features has benefit over single scale, non aggregated cases.

4.3.1 Evaluation Methodology

Metrics. We compare the number of predicted customer trouble tickets and that of genuine customer trouble tickets and measure the accuracy of prediction of service quality issues by false negative rate (FNR) and false positive rate (FPR). The FNR and FPR are computed per user basis.

$$\begin{aligned} FNR &= \frac{\text{\#of time bins that Q-score fails to predicts a trouble ticket}}{\text{\#of time bins that have genuine trouble tickets}} \\ FPR &= \frac{\text{\#of time bins that Q-score incorrectly predicts a trouble ticket}}{\text{\#of time bins that do not have any trouble ticket}} \end{aligned}$$

Note that due to the sparsity in the occurrence of user feedback (i.e., trouble tickets), the number of time bins without any use feedback is orders of magnitude higher than the number of time bins with user feedback.

Training and testing sets. In our evaluation of Q-score system, we use data sets collected from a commercial IPTV network provider in US over two months time period from August 1st, 2010 to September 30th, 2010. Unless otherwise mentioned, we use 15 days of data collected from August 15th, 2010 to August 29th, 2010 as the training data set for β and the subsequent 15 days of data collected from September 1st, 2010 to September 15th, 2010 as the testing data set. In addition, we use multi-scale temporal aggregation of $X_{Temp.Comb.}$ combining δ of 3-24 hours and multi-scale spatial aggregation of

$X_{Spat.Comb.}$ combining spatial levels of user, DSLAM, CO, and VHO as the default setting. Lastly, we set the default feedback time bin γ to be $\gamma = 24$ hours.

We assign λ a small positive value within $(0, 0.05]$. While different λ exhibit small difference in accuracy, the optimal λ varied from dataset to dataset. Since the selection of λ is specific to dataset in each test, we present the results with the best λ while omitting to show its actual value.

4.3.2 Results

4.3.2.1 Accuracy Analysis

We begin our evaluation by assessing how well Q-score follows the ground truth of user-perceived service quality. In our evaluation, we use user feedback as approximation of the ground truth of user-perceived service quality issues in training and testing Q-score system. Recall that the user feedback is incomplete in reflecting user perceived service quality. In fact, the user feedback captures a subset of user perceived service quality issues and thus underestimates the actual occurrences of service performance degradation. Fortunately, major and/or long lasting service performance degradation are likely to be captured by the user feedback [81]. Hence, it is likely that the computed Q-score underestimates the actual user perceived performance issues, but expected to capture major outages and performance degradation.

While Q-score does not perfectly match with the user perceived service quality at the individual user level, the change or trend in the distribution of

Q-score are expected to follow closely with that of the actual service quality degradation at certain spatial aggregation level. In our evaluation, we choose CO as the aggregation level¹. By summing up individual users' feedback within each CO into a single value, we obtain an aggregation vector $\mathbf{S}_{\text{actual}}$ of user feedback. Since $\mathbf{S}_{\text{actual}}$ is a spatio-temporal aggregation of user feedback, its element now signifies the level of user-perceived service quality issues. Similarly, by summing up the individual users' Q-score inside each CO into a single value, we can obtain an aggregation vector of Q-scores $\mathbf{S}_{\text{estim}}$ that signifies our estimated level of user-perceived service quality.

To evaluate the significance of the relation between the actual ($\mathbf{S}_{\text{actual}}$) and estimated ($\mathbf{S}_{\text{estim}}$) user perceived service quality level, we run an F-test between them. Let the null hypothesis $H_0 : r = 0$ where $\mathbf{S}_{\text{actual}} = r * \mathbf{S}_{\text{estim}}$. We find that for the significance level of 0.1, the hypothesis test is rejected, implying that the relation between the two vectors does exist. A Pearson's correlation test also shows relatively high correlation coefficients between $\mathbf{S}_{\text{actual}}$ and $\mathbf{S}_{\text{estim}}$, proving that the relationship between the two is linear. In other words, Q-score does follow the user-perceived service quality.

Because CO level aggregation represents spatial proximity of user geographical locations, user feedback rates can be different across COs. To evaluate if CO aggregation introduce any bias on the results, we also conduct the

¹We considered various levels of spatial granularity in the IPTV hierarchy including DSLAM, CO, and VHO levels. Among them, CO level aggregation is most adequate for the accuracy analysis because it yields statistically sound number of user IDs in each CO and enough number of COs to make meaningful comparisons between aggregation vector Ses .

Aggregation method	P value in F-test	Correlation coefficient
CO	0.00	0.6826
Random	2.21e-31	0.7165

Table 4.2: Accuracy analysis results of Q-score

same evaluation using a random grouping with the same number of groups as the number of COs and compute aggregation vectors. Table 4.2 summarizes F-test and Pearson’s correlation tests results for both CO level aggregation and random grouping based aggregation. The random grouping based aggregation generally shows the same results as the CO level aggregation, supporting that Q-score indeed follows user feedback regardless of how we aggregate users in Q-score computation.

4.3.2.2 Multi-scale Temporal Aggregation

In this section, we evaluate the impact of different time-bin size (δ) on network indicators (single-scale temporal level aggregation). Then we show the performance benefits by using multi-scale temporal aggregation on network performance indicators (multi-scale temporal level aggregation).

Figure 4.3 shows the Q-score on FPR-FNR trade-off curves using various δ s ranging from 3 hours to 24 hours (*i.e.*, each curve corresponds to an \mathbf{X} with a given δ). Note that FPR shown on the X-axis is in log-scale and FPR shown on Y-axis is in normal scale. The figure shows that the prediction accuracy gets generally better as we shorten δ (*i.e.*, the curve gets closer to the lower left corner of the plot). However, comparing $\delta = 3hours$ and

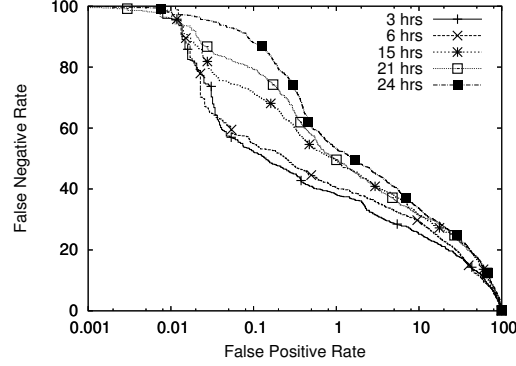


Figure 4.3: Comparison of various δ s on features (performance indicators)

$\delta = 6\text{hours}$, their FNR overlaps over different range of FPR, indicating that there is no single optimal δ to be chosen.

Figure 4.4 shows the results of $\mathbf{X}_{\text{Temp.Comb.}}$ by applying multi-scale temporal aggregation on network performance indicators. There are three curves obtained by combining (i) shorter time bins of 3-12 hours, (ii) longer time bins of 15-24 hours, and (iii) the entire range of 3-24 hours. We observe that (iii) provides the best performance among them. At the same time, (iii) is also strictly better than any curves in Figure 4.3, proving that multi-scale temporal aggregation on network performance indicators does improve the accuracy of Q-score prediction on service quality issues.

4.3.2.3 Multi-scale Spatial Aggregation

We now evaluate the impact of various levels of spacial aggregation on network performance indicators and the benefit of using multi-scale spatial aggregation in Q-score.

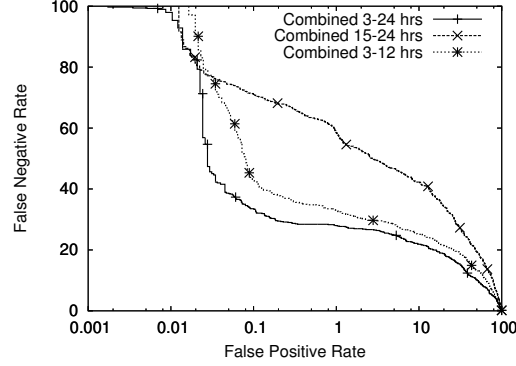


Figure 4.4: Comparison of multi-scale temporal aggregations on features (performance indicators)

Figure 4.5 shows the trade-off curves of \mathbf{X} with various single-scale spatial aggregation ranging from user ID ($\mathbf{X}_{\text{userID}}$), to DSLAM ($\mathbf{X}_{\text{DSLAM}}$), to CO (\mathbf{X}_{CO}), and to VHO (\mathbf{X}_{VHO}) level. As the spatial aggregation level changes from user ID to DSLAM (*i.e.*, smaller-sized region to larger-sized region), we observe that the FNR increases from 35% to 100% when FPR is at 0.1%. A possible explanation to this is that if the service quality issues reported by users are more related to home network side problem rather than the provider network problem, spatial aggregation of network performance indicators can attenuate signals relevant to the individual users at home network side. As we will show in Section 4.4.1, by analyzing significant KPIs, we are able to confirm that the significant KPIs are mostly related to STB and RG (*i.e.*, home network devices) while backbone network is suggested to be well provisioned. In addition to the single-scale spatial aggregation, the first plot of Figure 4.5 (denoted as ‘USER + DSLAM + CO + VHO’) shows multi-scale spatial aggregation (with measurement matrix $\mathbf{X}_{\text{Spat.Comb.}}$). We observe that

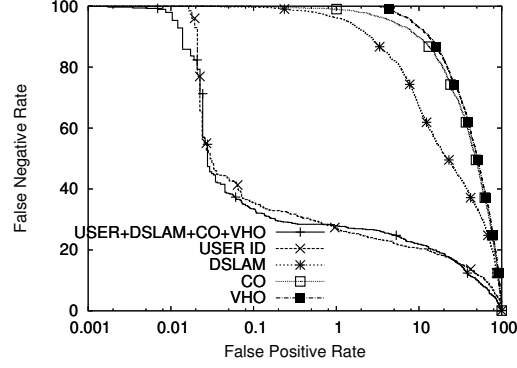


Figure 4.5: Comparison of various spatial aggregation levels on features (performance indicators)

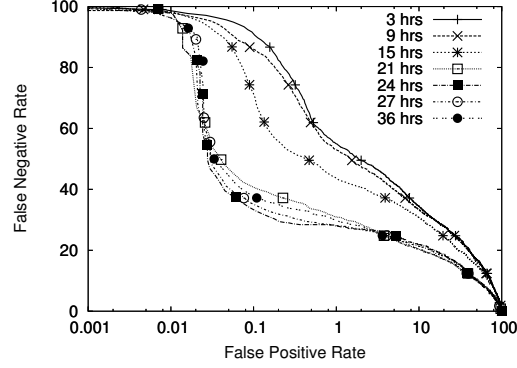


Figure 4.6: Comparison of various time bin size γ on user feedback

the multi-scale spatial aggregation outperforms any single-scale aggregation in terms of overall prediction accuracy, proving that the regression algorithm makes the most accurate selection of spatial level of features.

4.3.2.4 Feedback Aggregation

To show the effect of user feedback duration being aggregated together, Figure 4.6 compares various lengths of γ . We observe that as γ gets longer, the

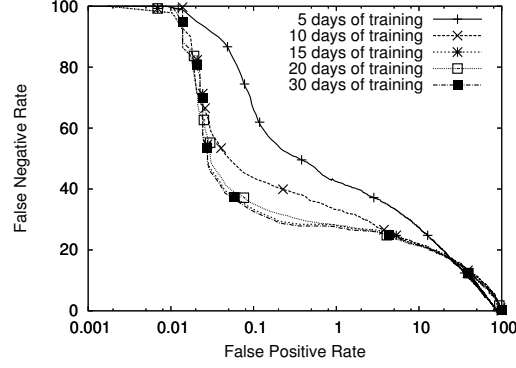


Figure 4.7: Comparison of accuracy with various training durations

regression performance gets better. An explanation for this is, as mentioned in Section 4.2.3, there is a significant delay between the occurrence of a problem and the filing of user feedback. Due to the elongated delay, time-bins with short γ s may fail to contain feedback correlated with significant network indicator values.

4.3.2.5 Sensitivity to Training Duration

Finally, we evaluate the sensitivity of testing accuracy on the duration of training. In this experiment, we fix the testing duration and assess how accuracy changes by varying the training duration. Table 4.3 shows the dates of training and testing periods used in our evaluation. Figure 4.7 shows the accuracy trade-off curves of using different training durations. We observe that in general, the testing accuracy improves as we increase the training duration. However, the gain becomes marginal once the training duration is longer than 15 days. This result suggests that using 15 days as training period is a good

	Duration	Dates
Testing duration	15 days	09/01/2010 - 09/15/2010
Training durations	5 days	08/25/2010 - 08/29/2010
	10 days	08/20/2010 - 08/29/2010
	15 days	08/15/2010 - 08/29/2010
	20 days	08/10/2010 - 08/29/2010
	30 days	08/01/2010 - 08/30/2010

Table 4.3: Training and testing durations

choice.

A closer examination on the curves corresponding to the use of 15 and 20 days of training duration reveals that the accuracy of using 15 days training duration is marginally better. Possible reason for this is that in the month of August, there was a network-wide STB firmware upgrade. The upgrade that took place between 08/10/2010 and 08/14/2010 could have obstructed measurement of STB logs (*i.e.*, STB audio and video quality measurement logs, syslog, reset and crash logs) and caused learning of β to be affected. Since this kind of glitches in data are common in practice, we take small amount of noise as granted. In all, we observe that 15 days of training is enough to learn β .

4.4 Application

In this section, we demonstrate the utility of Q-score by presenting three applications on it. First, we present a set of network KPIs that are closely related to user-perceived service quality. Second, we illustrate how much Q-score can predict user complains. Third, we show the possibility of intelligently

dimensioning workforce into troubling regions. In all the applications, we successfully identified interesting findings by running run-time analysis of Q-score on an ongoing basis.

4.4.1 Identification of Significant KPIs

A typical IPTV service already supports millions of user devices. If for every single device, few KPIs need to be monitored continuously, the measurement space can easily reach to the order of billions. In addition, time-lapse analysis in the diagnosis (as many diagnosis schemes employs) requires multiple measurements in short periods of time. Thus, in service assurance of a large-scale IPTV systems, it is infeasible to blindly measure, collect, and analyze such large volume of diverse KPIs from the entire network. In this application, we discuss about our experience on identification of a small number of significant KPIs with respect to user-perceived quality of experience.

Significant KPIs. In the generation of Q-score, we relate the network KPIs and user feedback by means of the factor β . β measures the relevance of significant KPIs by its magnitude. The analysis of the magnitude of β for different temporal aggregation levels indicates how KPIs correlate with user feedback. Tables 4.4 and 4.5 list top ten significant KPIs for relatively long history hours (15-24 hours) and short history hours (3-9 hours), respectively. Being regressed with individual users' feedback, the significant KPIs exhibit some commonality (shown in bold) as well as differences.

From the KPIs relevant to STB packet statistics, we observe that “tuner

fill”, “hole without session packets”, “bytes processed per sec” are particularly interesting KPIs. “Tuner fill” logs the number of packets lost by STBs before they are requested for retransmission. The lost packets are supposed to be retransmitted by D-Servers. Tuner fill counts can be related with video quality in that they indicate the condition of the delivery network and gives a sense of the average packet loss that would occur without any packet recovery scheme. A ‘hole’ represents time interval greater than a given threshold (assumed to affect video quality) in which no video packets have been received. ‘Hole without session packets’ counts the number of such holes occurred during a STB’s viewing session (since user’s last channel change).

On the audio and video related KPIs, “decoder error” logs a rather general type of errors occurred during decoding of audio data. Decoder errors can occur due to various situations including, but not limited to, out-of-order data packet reception, audio buffer underrun or overrun, and packet loss. ‘DRM errors’ and ‘crypto error’ indicates errors caused by video DRM decoder . This error can occur when encoder packets containing DRM keys are lost. In IPTV, every video program is encoded with DRM, and inability of decoding DRM blocks viewing of the programs. Thus, the occurrence of this error blocks TV viewing until new encoder keys are received regardless of receipt of the data packets. Lastly, there is ‘video frames dropped’ error which represents the number of video frames drops (below normal frame rate of 29.97 frames per second) due to packet loss or decoder errors. When large frame drop occurs, viewers can notice choppy or skippy motions. The identification of the

KPI Type	KPI Label	β Coef.
STB packet stat	RTP payload error	0.68
	Tuner fill	0.63
	Hole Too Large	0.61
	Decoder stall	0.42
	Bytes processed per sec	-0.32
Audio	Audio decoder errors	0.84
Video	Video DRM errors	0.73
	Video decoder errors	0.53
	Video frames decoded	-0.49
	Video data throughput	-0.49

Table 4.4: Significant KPIs for large δ (15-24 hrs)

KPI Type	KPI Label	β Coef.
STB packet stat	Hole without session packets	0.60
	Tuner fill	0.57
	Bytes processed per sec	-0.34
	ECM parse errors	0.32
Audio	Audio decoder errors	1.03
	Audio samples dropped	0.84
	Audio crypto error	0.64
	Audio data dropped	0.55
	Audio DRM errors	0.34
Video	Video DRM errors	0.63

Table 4.5: Significant KPIs for small δ (3-9 hrs)

important KPIs uncovers information that were missed out in controlled environments. In our case, we discovered that DRM errors is one of the most prominent indicators of video issues and added them to be considered in the simulated lab tests.

Observations. We observe an interesting finding by comparing significant KPIs of long-term event duration (*i.e.*, large δ) and short-term event duration

KPI Type	KPI Label	β Coef.
STB packet stat	Tuner fill	0.67
	Src unavailable received	0.5
	Hole without session packets	0.52
	ECM parse errors	0.35
	Bytes processed per sec	-0.33
Audio	Audio decoder errors	0.74
	Audio data dropped	0.57
	Audio crypto error	0.44
Video	Video DRM errors	0.68
	Video frames dropped	0.65

Table 4.6: Significant KPIs for multi-scale temporal aggregation (0-24 hrs)

(*i.e.*, small δ). The finding is that the former tend to have more video related KPIs as the most significant ones, whereas the latter has more KPIs related to audio. This relates with the relevance that audio has with respect to video in the user experience. Being the primary data, audio data is more susceptible to losses and errors than the secondary data, video. The reason is because the total volume of the data in audio is much less than that of the video, thus the impact of lost or delayed audio data is relatively greater than that of video data. Naturally, the viewers of the programs have less tolerance to audio issues than to video issues and complain about audio issues much earlier than video issues. The contrasting finding between long and short history hours have uncovered that, depending on the characteristics of the issues (*i.e.*, whether the issue is about audio or video), there are difference in urgency.

Another finding from the KPI analysis is drawn from multi-scale temporal aggregation. As shown in Table 4.6, by combining long-term and short-term event duration δ in regression, we observe both video and audio related

issues appear as the most significant KPIs. This further confirms the effectiveness of letting regression algorithm to choose important KPIs among multiple temporal aggregations.

Noticing that different KPIs have different degrees of relevancy to user feedback, we aim to guide monitoring of network KPIs by enlisting a small number of significant KPIs to user-perceived service quality. This way, forthcoming fine-grained network diagnosis can focus on the significant KPIs rather than analyzing excessive amount of KPIs.

4.4.2 Predicting Bad Quality of Experience

In order for Q-score to be useful for alerting services it should have the capability to provide triggers well before users start to complain. Thus, there is a need to study how much into the future we can infer user complaints using Q-score. To understand the feasible level of proactiveness in Q-score, we have evaluated two characteristics: (i) the growth pattern of Q-score over time and (ii) stability of Q-score with time gap between network events and user feedback.

Growth of Q-score over time. Figure 4.8 shows the growth pattern of Q-score for individual user IDs who filed trouble tickets. In the figure, we align the time by the trouble ticket filing time ($time = 0$) and observe how Q-score grows. The solid line represents the average value of the scores and the upper and lower tips of error bars represent one standard deviation plus and minus

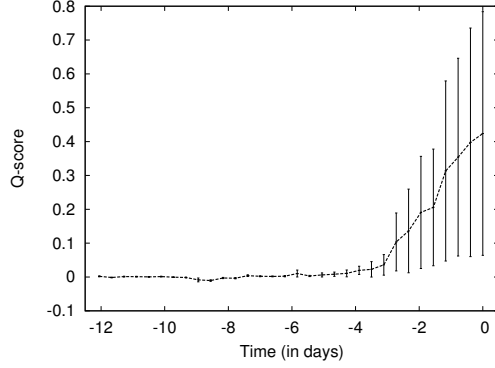


Figure 4.8: Growth pattern of Q-score

the average. From the graph, we observe that the increase of average Q-score is close to linear when it is greater than 0.05. The monotonic and gradual increase of Q-score suggests possibility of using Q-score as a proactive trigger for alerting because (i) it keeps increase once it becomes non-negligible level and (ii) its growth is not too abrupt. However, due to great variance among different users' Q-scores, we cannot use Q-score of 0.05 as the significant value triggering forthcoming actions. In stead, we seek for a more realistic lead time by conducting a further study on the stability of Q-score.

Feasible level of proactiveness. Next, we test for the stability of Q-score by skipping time interval between the occurrence times of network events b_i and user feedback c_i . The default time gap (or skipping interval) between $s_i + \delta$ and u_i is 0 hour because we set $u_i = s_i + \delta$ in Section 4.2.3. In this test, we add time gap τ to the equation $u_i = s_i + \delta + \tau$. By increasing τ in online monitoring step of Q-score generation, we test for the stability of Q-score in proactive, early warning.

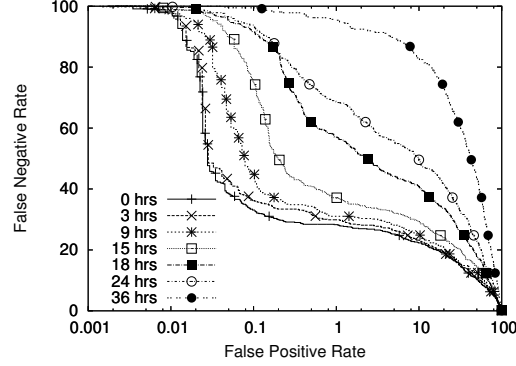


Figure 4.9: Comparison of accuracy of Q-scores with different skip intervals

With various τ ranging from 0 hours to 36 hours, Figure 4.9 exhibit FPR-FNR of learned β with different skipping times. As we increase τ , [5 we observe that FPR-FNR trade off gets worse. While the choice of lead time should mainly left to the discretion of network administrators, we find 9 hours of lead time is at the feasible level observing 9 hours of skip interval preserves 0.1% of FPR only sacrificing 10% of FNR (*i.e.*, FNR is 30% when skip interval is 0 hours and 40% when skip interval is 9 hours).

4.4.3 Dimensioning Customer Care Workforce

If network problems occur locally to regional service areas rather than globally impacting the entire service regions at once, an efficient management of field operators (*e.g.*, customer care representatives and repair men at customer premises) and servicing resources (*e.g.*, devices for fine-grained monitoring of network) would be to dynamically allocate them to challenging service regions than assigning static work areas. Thus, predicting the volume of fore-

coming issues to a service region at a given time is beneficial in adaptively allocating workforce across service regions. In this application, we assess the possibility of pre-allocating customer care workforce to potentially troubling service areas using Q-score. To begin, we first assess the volume of service quality issues per different spatial regions and see if the issues are contained locally or spread out globally.

Spatial distribution of user feedback. Figure 4.10 shows spatial distribution of user feedback across different COs. x -axis shows indexes of different COs, z -axis shows temporal trend. y -axis shows the percentage of user complaints (*e.g.*, value of 1 represents that 1% of users in the CO have complained on a given time). At a given time, we observe that high user feedback is local to each CO. And over time, the areas of high user feedback changes from one CO to another. From the fact that high feedback values generally being uncorrelated across time and CO (or space), we can affirm that the issues are temporal rather than permanent and local to an area rather than being global.

Leveraging Q-score for dimensioning workforce. Now that we have seen the possibility of dynamic resource allocation over different COs, we evaluate how closely Q-score follows user feedback in its magnitude when aggregated across individuals within each COs. Note that, to focus on its similarity to user feedback rate, we ignored the lead time of Q-score in this test. Figure 4.11 shows the trend of Q-score and user feedback aggregated per-CO. In doing so, Q-scores of individual user ID are first computed, and the scores corresponding to individuals within each CO are aggregated together to form per-CO Q-score.

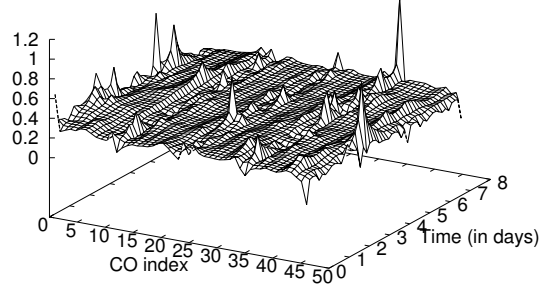


Figure 4.10: Percentage of users with trouble tickets over different spatial locations (COs) and times.

Figure 4.11 (a) shows the trend of per-CO Q-score and user feedback for a CO with relatively high customer feedback (*i.e.*, complaints). Over the course of 24 days, the percentage of users with complaints shown on the y -axis gets as high as 11%. Despite that there are some overestimations, the general trend of per-CO Q-score closely follows that of user feedback. Figure 4.11 (b) shows per-CO Q-score and user feedback for COs with moderately high complaints. We again see that the Q-score follows feedback whenever feedback increases over 2%. Figure 4.11 (c) shows the same for a CO with low customer complaints. Because there are only small increase (2% of users complaining) in the user feedback, Q-score remains at low level of 0.17% on average. From the observations from three different COs with high, medium, and low level of feedback, we confirmed that Q-score, when aggregated across individuals within each CO, closely follows the trend of per-CO user feedback. Since Q-score is confirmed to have several hours of lead time before users begin to

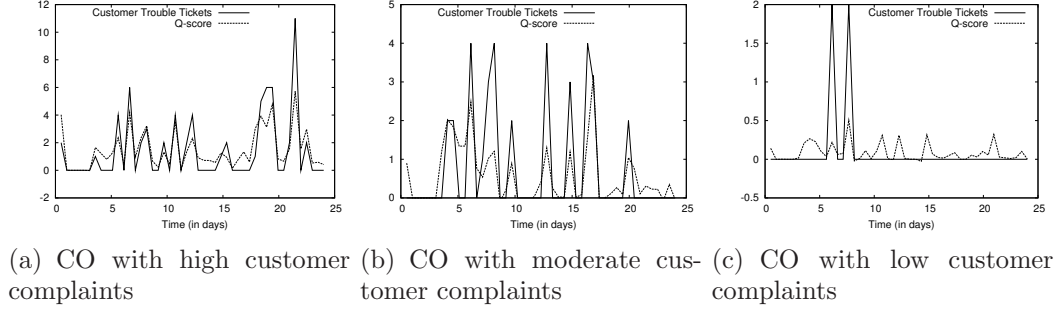


Figure 4.11: Trend of customer complaints and Q-score per CO.

complain, we can leverage Q-score in dimensioning workforce and prioritizing resources to areas with more upcoming issues ahead of time.

4.5 Summary of Proactive Service Quality Assessment

In this chapter, we develop Q-score, a novel framework for proactive assessment of user perceived service quality in large operational IPTV network. By associating coarse-grained network KPIs with imperfect user feedback, Q-score generates a single score that represents user-perceived quality of experience (QoE). Accuracy analysis of Q-score reveals that it is able to predict 60% of service problems reported by customers with only 0.1% of false positive rate. Applying Q-score to various application scenarios, we have: (i) identified a set of KPIs most relevant to user-perceived quality of experiences; (ii) quantified how early it can alert bad quality of experience; (iii) observed possibility to pre-allocate customer care workforce to potentially troubling service areas.

There are many other network services that are sensitive to service quality but lacks objective measures of user-perceived quality of experience.

Our future work include applying the proactive service quality assessment beyond the specific context of IPTV network. For example, we plan to apply Q-score to VoIP and mobile networks so that operation teams can predict user distressing call drops and voice quality degradation without having to wait for customers to complain.

Chapter 5

Conclusion

This chapter summarizes and discusses our contributions in Section 5.1, and point out directions for future research in Section 5.2, both in addressing the limitations of this work and in pursuing new avenues.

5.1 Contributions

In this thesis, we endeavored to present network path analytics that are scalable to be ran on massive networks, flexible to accommodate various objectives from different administrative tasks, and robust to dynamic changes in the networks. The quadratic growth in the number of network paths with respect to the number of network nodes poses significant challenges.

Addressing the challenges, we made the following contributions for different analysis conducted on various types of networks.

5.1.1 Accurate and Flexible Inference of Network Properties

We first design, implement, and evaluate an algebraic approach to *exact reconstruction of network path properties*. For an overlay of n end hosts, we selectively monitor a basis set of $O(n \log n)$ paths which can fully describe all

the $O(n^2)$ paths. Then the measurements of the basis set are used to infer the loss rates of all other paths. Our approach works in real time, offers fast adaptation to topology changes, distributes balanced load to end hosts, and handles topology measurement errors. Both simulation and real Internet implementation yield promising results.

Then in *approximate reconstruction of network path properties*, we present a flexible framework for large-scale network measurement and inference which further reduces the overhead of active measurement in exchange of slight accuracy sacrifice. The framework consists of two major components: the design of measurement experiments, and network inference. For the former, we leverage powerful tools developed in the field of Bayesian experimental design. For the latter, we build on top of a number of existing network tomography techniques to infer network properties based on partial, indirect observations. Our framework is flexible, and can accommodate a variety of design objectives, such as differentiated, augmented, and multi-user designs. It is also highly scalable and can design measurement experiments that span thousands of routers and end hosts.

5.1.2 Scalable and Accurate Social Network Analysis

In our first step towards node proximity approximation, *scalable proximity inversion*, we develop two novel techniques, proximity embedding and proximity sketch, to approximate proximity measures that have a low-rank structure, such as Katz measure, in massive, highly dynamic online social

networks. Our techniques are accurate and can easily handle networks with millions of nodes, which are several orders of magnitude larger than what existing methods can support.

Since the first scheme cannot approximate proximity measures that are not low-rank, in the subsequent work, *clustered spectral graph embedding (CSGE)*, we develop a novel dimensionality reduction technique that can embed a massive original graph into a much smaller graph. The embedded graph captures the essential clustering and spectral structure of the original graph and allows a wide range of analysis tasks to be performed in an efficient and accurate fashion.

To evaluate the performance of clustered spectral graph embedding, we explore three important social network analysis tasks (proximity estimation, missing link inference, and link prediction) using three large-scale real-world social network datasets (Flickr, LiveJournal and MySpace; with up to 2 million vertices and 90 million edges). For proximity estimation, clustered spectral graph embedding achieves up to an order of magnitude improvement in efficiency and up to several orders of magnitude improvement in accuracy (when the proximity measures of interest have high intrinsic dimensionality). For missing link inference and link prediction, our technique consistently yields better accuracy across different social networks. These results clearly demonstrate the effectiveness and the potential values of our approach.

5.1.3 Proactive Service Quality Assessment

In application service networks, we develop a novel framework for proactive assessment of user perceived service quality in large operational IPTV network. By associating coarse-grained network KPIs with imperfect user feedback, Q-score generates a single score that represents user-perceived quality of experience (QoE). Accuracy analysis of Q-score reveals that it is able to predict 60% of service problems reported by customers with only 0.1% of false positive rate. Applying Q-score to various application scenarios, we have: (i) identified a set of KPIs most relevant to user-perceived quality of experiences; (ii) quantified how early it can alert bad quality of experience; (iii) observed possibility to pre-allocate customer care workforce to potentially troubling service areas.

There are many other network services that are sensitive to service quality but lacks objective measures of user-perceived quality of experience. Our future work include applying the proactive service quality assessment beyond the specific context of IPTV network. For example, we plan to apply Q-score to VoIP and mobile networks so that operation teams can predict user distressing call drops and voice quality degradation without having to wait for customers to complain.

5.2 Future Works

There are several avenues for future work that we plan to extend the results of this thesis. First, in the path performance measurement and inference

of IP network, we plan to further enhance the robustness of our experimental design and inference by making the design *fault tolerant*. Specifically, we want a design that minimizes the *worst-case* design criterion in the presence of multiple faulty paths (*i.e.*, paths that experience either failures or major routing changes). We also are interested in applying our techniques to estimating other network properties, such as traffic matrix estimation. In particular, we may use Bayesian experimental design to identify strategic locations to place additional measurement capabilities to enhance the accuracy of traffic matrix estimation. Then, we would like to extend our framework to incorporate additional design constraints and to handle non-linear metrics by leveraging the existing work on non-linear Bayesian experimental design [13]. Finally, we are interested in developing light-weight techniques to obtain better prior information.

Second, in the social network analysis, we envision to further expand the application of clustered spectral graph embedding. One expansion is to community detection. Applying the results of clustering to infer the formation of grouping among nodes, we expect to gain insights on both the effectiveness and possible improvements to normalized cut based co-clustering methods. The other expansion can be done by applying the dimensionality reduction scheme to discovery of user identity across multiple online social networks. By comparing embedded graphs of multiple online social networks rather than the original network topologies, we can develop ways to concisely represent characteristics of those well-connected nodes in massive-scale online social networks.

With the knowledge of the well-connected nodes in each network, we can then de-anonymize a subset of users that appear across different networks as it is done in [73].

Third, expanding the results of service quality assessment, we can devise better service assurance in cloud computing, data center, and application networks. Today's network platforms such as cloud computing operates hundreds of thousands of servers and storages in their data centers, therefore, it is challenging to obtain a holistic view that effectively manages the entire system. Classic network monitoring only has partial view of the whole system, *i.e.*, they only assure reliability of individual network elements, network segments, sub-networks, or the routing among components. A service assurance should integrate the network performance with service quality and troubleshoot network faults that directly impact the customer-perceived service quality. We plan to first assess the service quality in the large-scale networks and obtain a set of key network performance metrics. We will then initiate fine grained analysis on a small number of targeted devices. In the analysis and troubleshooting, it is imperative to devise scalable, flexible, and robust methodologies. We plan to explore multi-resolution techniques and spatio-temporal aggregation for effective handling of such ultra-large scale networks.

Bibliography

- [1] <http://www.bittorrent.com>.
- [2] A. Abou-Rjeili and G. Karypis. Multilevel algorithms for partitioning power-law graphs. In *IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2006.
- [3] Alexa global top 500 sites. http://www.alexa.com/site/ds/top_sites.
- [4] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *18th ACM SOSP*, 2001.
- [5] D. G. Andersen et al. Resilient overlay networks. In *Proc. of ACM SOSP*, 2001.
- [6] E. Anderson et al. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.
- [7] Arpack. <http://www.caam.rice.edu/software/ARPACK/>.
- [8] A. L. Barabasi, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek. Evolution of the social network of scientific collaboration. *Physica A: Statistical Mechanics and its Application*, 2002.
- [9] R. M. Bell, Y. Koren, and C. Volinsky. Chasing \$1,000,000: How we won the Netflix Progress Prize. *Statistical Computing and Statistical Graphics Newsletter*, 18(2):4–12, 2007.
- [10] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 7(10), 2008.
- [11] T. Bu, N. Duffield, F. Presti, and D. Towsley. Network tomography on general topologies. In *ACM SIGMETRICS*, 2002.

- [12] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.
- [13] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10:273–304, 1995. <http://citeseer.ist.psu.edu/chaloner95bayesian.html>.
- [14] S. Chatterjee and A. S. Hadi. Influential observations, high leverage points, and outliers in linear regression. *Statistical Science*, Vol. 1:379–416, 1986.
- [15] Y. Chen, D. Bindel, and R. H. Katz. Tomography-based overlay network monitoring. In *ACM SIGCOMM Internet Measurement Conference (IMC)*, 2003.
- [16] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. In *SIGCOMM '04: Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 55–66, New York, NY, USA, 2004. ACM Press.
- [17] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. In *Proc. of ACM SIGCOMM*, 2004.
- [18] D. B. Chua, E. D. Kolaczyk, and M. Crovella. Efficient monitoring of end-to-end network properties. In *Proc. of IEEE INFOCOM*, Jul. 2004.
- [19] D. B. Chua, E. D. Kolaczyk, and M. Crovella. Efficient monitoring of end-to-end network properties. In *IEEE INFOCOM*, 2005.
- [20] M. A. Clyde. Experimental design: A Bayesian perspective. *International Encyclopedia Social and Behavioral Sciences*, Apr. 2001.
- [21] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 2005.
- [22] C. Cortes, D. Pregibon, and C. T. Volinsky. Communities of interest. *Intelligent Data Analysis*, 2002.
- [23] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. New York: Wiley, 1991.

- [24] J. Davidsen, H. Ebel, and S. Bornholdt. Emergence of a small world from local interactions: Modeling acquaintance networks. *Physical Review Letters*, 2002.
- [25] T. Davis. LDL: A sparse LDL' factorization and solve package (version 1.2), 2005. <http://www.cise.ufl.edu/research/sparse/ldl/>.
- [26] J. Demmel. *Applied Numerical Linear Algebra*. SIAM, 1997.
- [27] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(11):1944–1957, 2007.
- [28] Digg. <http://www.digg.com>.
- [29] D. Donoho. For most large underdetermined systems of linear equations, the minimal l_1 -norm near-solution approximates the sparsest near-solution. <http://www-stat.stanford.edu/~donoho/Reports/2004/l1l0approx.pdf>.
- [30] D. Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 52(4):1289–1306, Apr. 2006.
- [31] D. Epema, M. Livny, R. van Dantzig, X. Evers, and J. Pruyne. A world-wide flock of Condors: Load sharing among workstation clusters. *Future Generation Computer Systems*, 1996.
- [32] Facebook. <http://www.facebook.com>.
- [33] Facebook statistics. <http://www.facebook.com/press/info.php?statistics>.
- [34] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationship of the Internet topology. In *ACM SIGCOMM*, 1999.
- [35] V. Fedorov. *Theory of Optimal Experiments*. Academic Press, New York, 1972.
- [36] Flickr. <http://www.flickr.com>.
- [37] S. Garriss, M. Kaminsky, M. J. Freedman, B. Karp, D. Mazieres, and H. Yu. RE: Reliable Email. In *Proc. of Networked Systems Design and Implementation (NSDI)*, 2006.
- [38] G. Golub and C. V. Loan. *Matrix Computations*. The Johns Hopkins University Press, 1989.

- [39] G. H. Golub, V. Klema, and G. W. Stewart. Rank degeneracy and least squares problems. Technical Report CS-TR-76-559, Stanford University, Aug. 1976. <http://www-db.stanford.edu/TR/CS-TR-76-559.html>.
- [40] G. H. Golub and C. F. V. Loan. *Matrix Computations*. Johns Hopkins University Press, Baltimore, Maryland, 2nd edition, 1989.
- [41] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, Baltimore, MD, third edition, 1996.
- [42] R. Govindan and H. Tangmunarunkit. Heuristics for Internet map discovery. In *IEEE INFOCOM*, 2000.
- [43] L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 11(9):1074–1085, Sep 1992.
- [44] P. C. Hansen. *Rank-Deficient and Discrete Ill-Posed Problems: Numerical Aspects of Linear Inversion*. SIAM, 1997.
- [45] S. Hill, F. Provost, and C. Volinsky. Network-based marketing: Identifying likely adopters via consumer networks. *Statistical Science*, 2006.
- [46] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, Vol. 12, No. 1:55–67, 1970.
- [47] E. M. Jin, M. Girvan, and M. E. J. Newman. The structure of growing social networks. *Physical Review Letters*, 2001.
- [48] B. Jones, D. Lin, and C. Nachtsheim. Bayesian D-optimal supersaturated designs, 2004. Submitted for publication.
- [49] L. Katz. A new status index derised from sociometric analysis. *Psychometrika*, 1953.
- [50] V. Krebs. Mapping networks of terrorist cells. *Connections*, 24(3):43–52, Winter 2002.
- [51] R. Larsen. Lanczos bidiagonalization with partial reorthogonalization. Technical Report DAIMI PB-357, Department of Computer Science, Aarhus University, 1998.

- [52] R. Lehoucq, D. Sorensen, and C. Yang. *Arpack Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia, 1998.
- [53] R. B. Lehoucq and D. C. Sorensen. Deflation techniques for an implicitly restarted arnoldi iteration. *SIAM Journal on Matrix Analysis and Applications*, 17(4):789–821, 1996.
- [54] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *CoRR*, abs/0810.1355, 2008.
- [55] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 695–704, New York, NY, USA, 2008. ACM.
- [56] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Proc. of Conference on Information and Knowledge Management (CIKM)*, 2003.
- [57] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *J. Am. Soc. Inf. Sci. Technol.*, 2007.
- [58] H. Lim, J. Hou, and C. H. Choi. Constructing Internet coordinate system based on delay measurement. In *Proc. of IMC*, 2003.
- [59] F. Lin and W. W. Cohen. Power iteration clustering. In *Proc. of ICML*, 2010.
- [60] D. V. Lindley. *Bayesian Statistics – A Review*. SIAM, Philadelphia, PA, 1972.
- [61] LiveJournal. <http://www.livejournal.com>.
- [62] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson. Inferring link weights using end-to-end measurements. In *ACM SIGCOMM Internet Measurement Workshop*, Marseilles, France, November 2002.
- [63] Y. Mao and L. K. Saul. Modeling distances in large-scale networks by matrix factorization. In *Proc. of IMC*, New York, NY, USA, 2004.

- [64] A. Medina, I. Matta, and J. Byers. On the origin of power laws in Internet topologies. In *ACM Computer Communication Review*, Apr. 2000.
- [65] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: Existing techniques and new directions. In *Proc. of ACM SIGCOMM*, Aug. 2002.
- [66] A. J. Miller and N. K. Nguyen. A Fedorov exchange algorithm for D-optimal design. *Applied Statistics*, 1994.
- [67] A. Mislove, K. P. Gummadi, and P. Druschel. Exploiting social networks for Internet search. In *Proc. of HotNets-V*, 2006.
- [68] A. Mislove, H. S. Koppula, K. Gummadi, P. Druschel, and B. Bhattacharjee. Growth of the flickr social network. In *Proc. of SIGCOMM Workshop on Social Networks (WOSN)*, 2008.
- [69] T. J. Mitchell. An algorithm for the construction of D-optimal designs. *Technometrics*, 1974.
- [70] MySpace. <http://www.myspace.com>.
- [71] MySpace 400 millionth user. <http://profile.myspace.com/index.cfm?fuseaction=user.viewprofile&friend%id=400000000>.
- [72] E. M. Nahum, C. C. Rosu, S. Seshan, and J. Almeida. The effects of wide-area conditions on WWW server performance. In *Proc. of SIGMETRICS*, Jun. 2001.
- [73] A. Narayanan and V. Shmatikov. De-anonymizing social networks. *Security and Privacy, IEEE Symposium on*, 0:173–187, 2009.
- [74] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 2003.
- [75] T. E. Ng and H. Zhang. Predicting internet network distance with coordinate-based approaches. In *Proc. of INFOCOMM*, 2002.
- [76] N. K. Nguyen and A. J. Miller. A review of exchange algorithms for constructing discrete D-optimal designs. *Computational Statistics and Data Analysis*, 1992.

- [77] Nielson Online. Fastest growing social networks for September 2008. http://blog.nielsen.com/nielsenwire/wp-content/uploads/2008/10/pr%ess_release24.pdf.
- [78] V. N. Padmanabhan, L. Qiu, and H. Wang. Server-based inference of Internet performance. In *Proc. of IEEE INFOCOM*, Mar. 2003.
- [79] J. A. Patel, I. Gupta, and N. Contractor. JetStream: Achieving predictable gossip dissemination by leveraging social network principles. In *Proc. of IEEE Network Computing and Applications (NCA)*, 2006.
- [80] PlanetLab. <http://www.planet-lab.org/>.
- [81] T. Qiu, J. Feng, Z. Ge, J. Wang, J. Xu, and J. Yates. Listen to me if you can: Tracking user experience of mobile network on social media. In *IMC*, 2010.
- [82] B. Recht, W. Xu, and B. Hassibi. Necessary and sufficient conditions for success of the nuclear norm heuristic for rank minimization. In *Proc. of 47th IEEE Conference on Decision and Control*, Dec. 2008.
- [83] S. Rhea, B. Godfrey, B. Karp, J. Kubiawicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. OpenDHT: A Public DHT Service and Its Uses. In *ACM SIGCOMM 2005*.
- [84] P. Sarkar, A. W. Moore, and A. Prakash. Fast incremental proximity search in large graphs. In *Proc. of ICML*, 2008.
- [85] M. Saunders. PDCO: Primal-Dual interior method for Convex Objectives, 2003. <http://www.stanford.edu/group/SOL/software/pdco.html>.
- [86] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 2000.
- [87] H. H. Song, T. W. Cho, V. Dave, Y. Zhang, and L. Qiu. Scalable proximity estimation and link prediction in online social networks. In *IMC '09: Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference*, pages 322–335, New York, NY, USA, 2009. ACM.
- [88] H. H. Song, T. W. Cho, V. Dave, Y. Zhang, and L. Qiu. Scalable proximity estimation and link prediction in online social networks. In *IMC*, 2009.

- [89] H. H. Song, A. Mahimkar, Z. Ge, J. Wang, J. Yates, and Y. Zhang. Q-score: Proactive service quality assessment in a large iptv system. Technical report, Department of Computer Science, The University of Texas at Austin, 2011. Manuscript submitted for publication.
- [90] H. H. Song, L. Qiu, and Y. Zhang. NetQuest: A flexible framework for large-scale internet measurement. In *ACM SIGMETRICS*, Saint-Malo, France, 2006.
- [91] H. H. Song, B. Savas, T. W. Cho, V. Dave, Z. Lu, I. Dhillon, Y. Zhang, and L. Qiu. Clustered embedding of massive social networks. Technical report, Department of Computer Science, The University of Texas at Austin, 2011. Manuscript submitted for publication.
- [92] N. Spring, R. Mahajan, and D. Wetherall. Measuring isp topologies with rocketfuel. In *ACM SIGCOMM*, 2002.
- [93] N. Spring, R. Mahajan, and D. Wetherall. Measuring ISP topologies with Rocketfuel. In *ACM SIGCOMM*, Pittsburg, USA, August 2002.
- [94] N. Spring, R. Mahajan, D. Wetherall, and H. Hagerstrom. Rocketfuel: An ISP topology mapping engine. <http://www.cs.washington.edu/research/networking/rocketfuel/>.
- [95] N. Spring, D. Wetherall, and T. Anderson. Scriptroute: A facility for distributed internet measurement. In *Proc. of USITS*, Mar. 2003.
- [96] L. Subrmanian, S. Agarwal, J. Rexford, and R. H.Katz. Characterizing the Internet hierarchy from multiple vantage points. In *IEEE INFOCOM*, 2002.
- [97] L. Tang and M. Crovella. Virtual landmarks for the Internet. In *Proc. of IMC*, 2003.
- [98] H. Tangmunarunkit et al. Network topology generators: Degree-based vs structural. In *ACM SIGCOMM*, 2002.
- [99] H. Tong, C. Faloutsos, and Y. Koren. Fast direction-aware proximity for graph mining. In *Proc. of KDD*, 2007.
- [100] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove. An analysis of social network-based Sybil defenses. In *Proc. of ACM SIGCOMM*, 2010.

- [101] D. Wagner and F. Wagner. Between min cut and graph bisection. In *MFC'S '93: Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*, pages 744–750, London, UK, 1993. Springer-Verlag.
- [102] L. Wang, K. Park, R. Pang, V. S. Pai, and L. Peterson. Reliability and security in the codeen content distribution network. In *USENIX*, 2004.
- [103] Wikipedia. Social network. http://en.wikipedia.org/wiki/Social_network.
- [104] Wikipedia. <http://www.wikipedia.org>.
- [105] YouTube. <http://www.youtube.com>.
- [106] H. Yu, M. Kaminsky, P. B. Gibbons, and A. Flaxman. SybilGuard: Defending against Sybil attacks via social networks. In *Proc. of SIGCOMM*, 2006.
- [107] Y. Zhang et al. On the constancy of Internet path properties. In *Proc. of SIGCOMM IMW*, 2001.
- [108] Y. Zhang, Z. Ge, A. Greenberg, and M. Roughan. Network anomography. In *Proc. Internet Measurement Conference*, Oct. 2005.
- [109] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg. Fast accurate computation of large-scale ip traffic matrices from link loads. In *Proc. of ACM SIGMETRICS*, Jun. 2003.
- [110] Y. Zhang, M. Roughan, C. Lund, and D. Donoho. An information-theoretic approach to traffic matrix estimation. In *Proc. of ACM SIGCOMM*, Aug. 2003.
- [111] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu. Spatio-temporal compressive sensing and Internet traffic matrices. In *Proc. of ACM SIGCOMM*, 2009.
- [112] Y. Zhao, Y. Chen, and D. Bindel. Towards unbiased end-to-end network diagnosis. In *Proc. of ACM SIGCOMM*, Sept. 2006.

Vita

Han Hee Song was born in Tokyo, Japan to Prof. Bang Ho Song and Prof. Sook Ja Lee. He graduated from Kyungpook National University's Attached High School, Daegu, Korea in 1997. He received the degree of Bachelor of Science in Computer Sciences from Yonsei University in August, 2004. Prior to graduation, he attended University of California, Berkeley for a year in 2003. From 1999 to 2001, he was employed at Dong-Suh Industrial Development Co., Ltd. in Seoul, Korea as a software engineer. In 2002 and 2003, he was employed at Soft-On-Net, Inc. in Seoul, Korea and Soft-On-Net Japan, Inc. in Tokyo, Japan as a software developer. He joined the University of Texas at Austin in Fall 2004 and received Master of Arts degree in Spring 2006 under supervision of Professor Yin Zhang and Lili Qiu. His master thesis was on scalable and flexible network measurement. He continued to study for his Ph.D. under supervision of Professor Yin Zhang. During his study, he interned at Hewlett-Packard Labs, Microsoft Research, and AT&T Labs - Research. Aimed at better understanding of network internals and efficient use of large networks, his Ph.D. dissertation is on measurement and inference, and analysis of large-scale network systems.

Permanent email address: hhsong@utexas.edu